Approximation Techniques bounded inference

COMPSCI 276, Spring 2011

Set 10: Rina Dechter

(Reading: Primary: Class Notes (8)
Secondary: , Darwiche chapters 14)

Probabilistic Inference Tasks

- Belief updating:

$$\mathbf{BEL(X_i) = P(X_i = x_i \,|\, evidence)}$$

- Finding most probable explanation (MPE)

$$\bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e)$$

- Finding maximum a-posteriory hypothesis

$$(a_1^*, \ldots, a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} P(\bar{x}, e)$$

$A \subseteq X$: **hypothesis variables**

- Finding maximum-expected-utility (MEU) decision

$$(d_1^*, \ldots, d_k^*) = \arg\max_{\bar{d}} \sum_{X/D} P(\bar{x}, e) U(\bar{x})$$

$D \subseteq X$: **decision variables**
$U(\bar{x})$: **utility function**

Finding $MPE = \max_{\bar{x}} P(\bar{x})$

Algorithm *elim-mpe* (Dechter 1996)

$\sum$ is replaced by ***max*** *:*

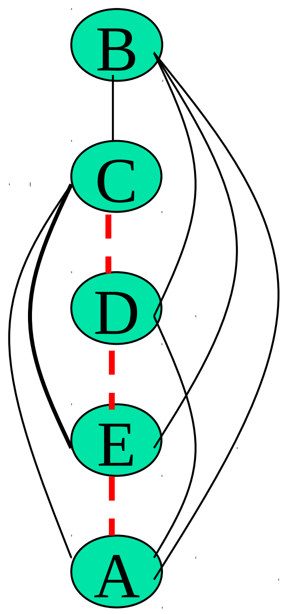
$$MPE = \max_{a,e,d,c,b} P(a)P(c\,|\,a)P(b\,|\,a)P(d\,|\,a,b)P(e\,|\,b,c)$$

$$\max_b \prod \longleftarrow \text{Elimination operator}$$

$$\begin{tikzcd}
\text{bucket } B: & \overbrace{P(b|a) \arrow[ddr]}^{\max_b \prod} & P(d|b,a) \arrow[dd] & P(e|b,c) \arrow[ddl] \\
\text{bucket } C: & P(c|a) \arrow[dr] & & \\
\text{bucket } D: & & \boldsymbol{h^C(a,d,e)} \arrow[d] & \\
\text{bucket } E: & e=0 \arrow[dr] & \boldsymbol{h^D(a,e)} \arrow[d] & \\
\text{bucket } A: & P(a) \arrow[d] & \boldsymbol{h^E(a)} \arrow[dl] & \\
& \boldsymbol{MPE} & &
\end{tikzcd}$$

W*=4

"induced width" (max clique size)



Generating the MPE-tuple

5. $b' = \arg\max_b P(b \mid a') \times P(d' \mid b, a') \times P(e' \mid b, c')$

4. $c' = \arg\max_c P(c \mid a') \times h^B(a', d', c, e')$

3. $d' = \arg\max_d h^C(a', d, e')$

2. $e' = 0$

1. $a' = \arg\max_a P(a) \cdot h^E(a)$

B: P(b|a) P(d|b,a) P(e|b,c)

C: P(c|a) $h^B(a,d,c,e)$

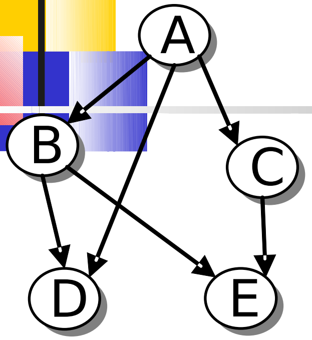
D: $h^C(a,d,e)$

E: e=0 $h^D(a,e)$

A: P(a) $h^E(a)$

Return (a', b', c', d', e')

Bucket Elimination



Query: $P(a \mid e=0) \propto P(a, e=0)$

Elimination Order: d,e,b,c

$$P(a,e=0) = \sum_{c,b,e=0,d} P(a)P(b \mid a)P(c \mid a)P(d \mid a,b)P(e \mid b,c)$$

$$= P(a)\sum_{c} P(c \mid a)\sum_{b} P(b \mid a)\sum_{e=0} P(e \mid b,c)\sum_{d} P(d \mid a,b)$$

Original Functions

D: $P(d \mid a,b)$

E: $P(e \mid b,c)$

B: $P(b \mid a)$

C: $P(c \mid a)$

A: $P(a)$

Messages

$\boxed{f_D(a,b)} = \sum_{d} P(d \mid a,b)$

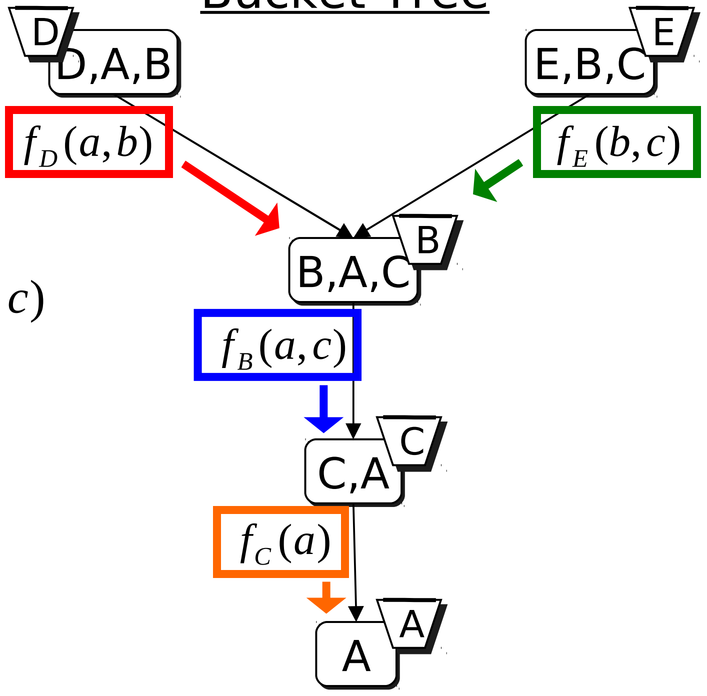
$\boxed{f_E(b,c)} = P(e=0 \mid b,c)$

$\boxed{f_B(a,c)} = \sum_{b} P(b \mid a) f_D(a,b) f_E(b,c)$

$\boxed{f_C(a)} = \sum_{c} P(c \mid a) f_B(a,c)$

$P(a,e=0) = p(A) f_C(a)$

Bucket Tree



Time and space exp(w*)

Approximate Inference

- Metrics of evaluation
- **Absolute error**: given e>0 and a query p= P(x|e), an estimate r has absolute error e iff |p-r|<e
- **Relative error**: the ratio r/p in [1-e,1+e].
- Dagum and Luby 1993: approximation up to a relative error is NP-hard.
- Absolute error is also NP-hard if error is less than .5

Mini-buckets: “local inference”

- Computation in a bucket is time and space exponential in the number of variables involved

- Therefore, partition functions in a bucket into “mini-buckets” on smaller number of variables

Mini-bucket approximation: MPE task

Split a bucket into mini-buckets =>bound complexity

$$\text{bucket}(X) = \underbrace{\{\, h_1, \dots, h_r, h_{r+1}, \dots, h_n \,\}}$$

$$h^X = \max_X \prod_{i=1}^{n} h_i$$

$$\underbrace{\{\, h_1, \dots, h_r \,\}} \qquad \underbrace{\{\, h_{r+1}, \dots, h_n \,\}}$$

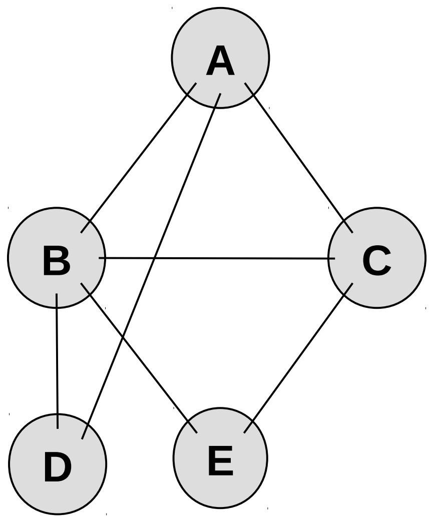
$$g^X = \left(\max_X \prod_{i=1}^{r} h_i\right) \cdot \left(\max_X \prod_{i=r+1}^{n} h_i\right)$$

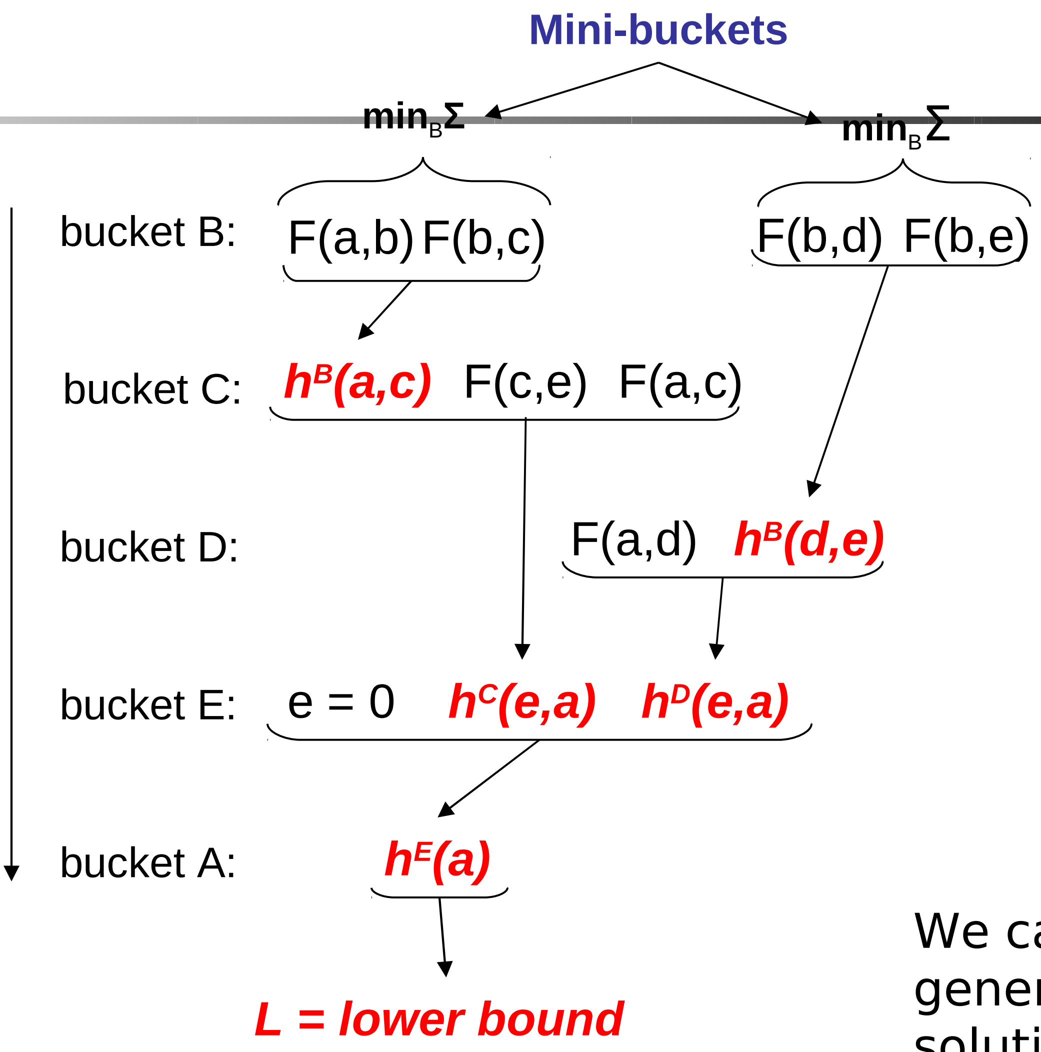
$$\downarrow$$

$$h^X \le g^X$$

Exponential complexity decrease : $O(e^n) \rightarrow O(e^r) + O(e^{n-r})$

Mini-Bucket Elimination





Mini-buckets

$\min_B \Sigma$ $\min_B \Sigma$

bucket B: $\underbrace{F(a,b)\ F(b,c)}$ $\underbrace{F(b,d)\ F(b,e)}$

bucket C: ***$h^B(a,c)$*** $F(c,e)$ $F(a,c)$

bucket D: $F(a,d)$ ***$h^B(d,e)$***

bucket E: $e = 0$ ***$h^C(e,a)$*** ***$h^D(e,a)$***

bucket A: ***$h^E(a)$***

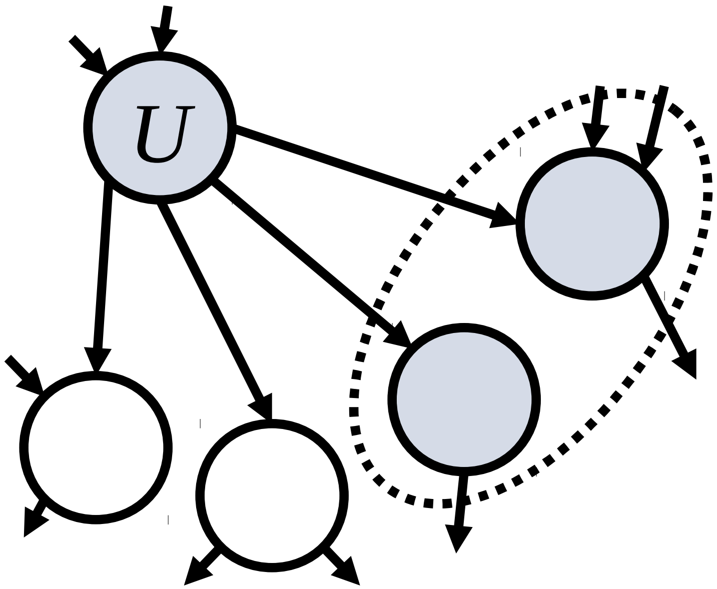
L = lower bound

We can generate a solution s going forward as before

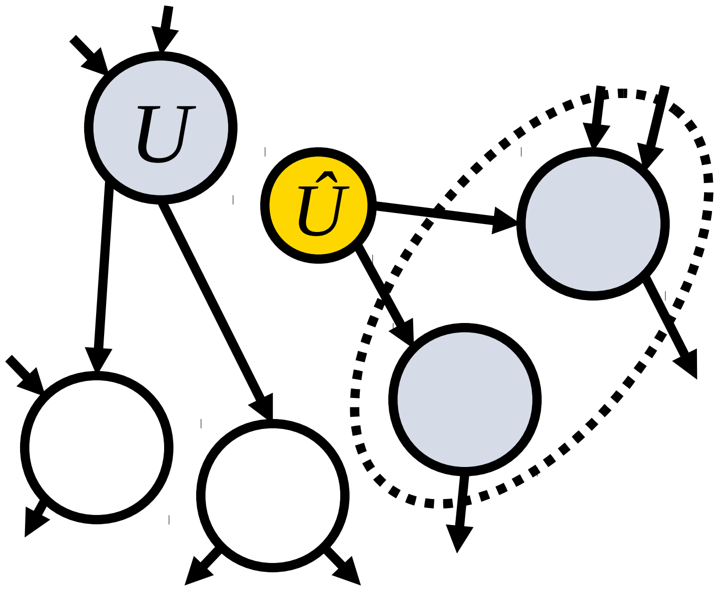
Semantics of Mini-Bucket: Splitting a Node

Variables in different buckets are renamed and duplicated
(Kask *et. al.*, 2001), (Geffner *et. al.*, 2007), (Choi, Chavira, Darwiche , 20

Before Splitting:
Network *N*



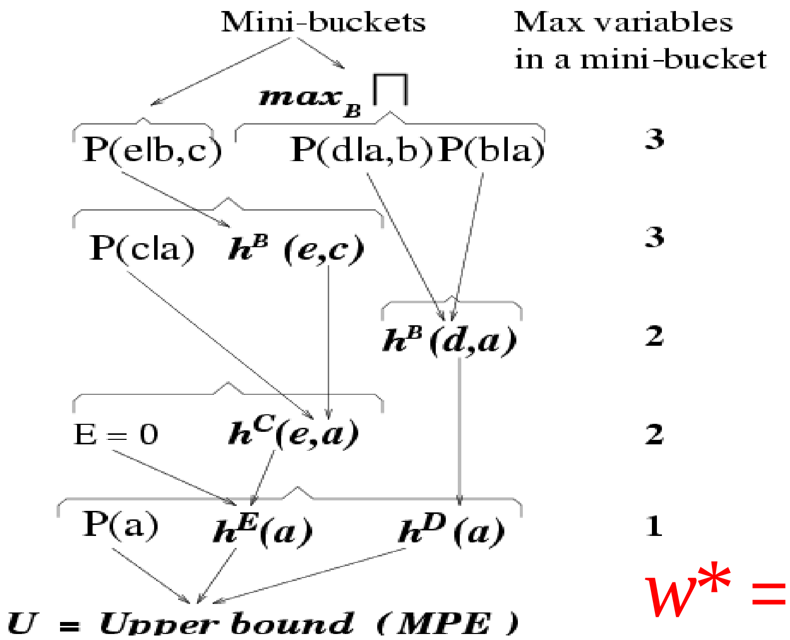
After Splitting:
Network *N*'



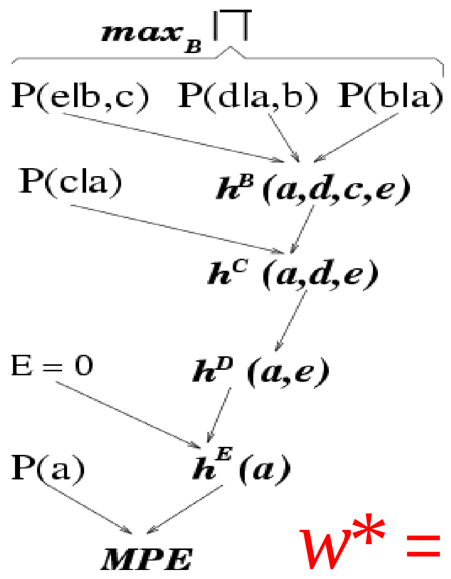
Approx-mpe(i)

- Input: i – max number of variables allowed in a mini-bucket
- Output: [lower bound (P of a sub-optimal solution), upper bound]

Example: approx-mpe(3) versus elim-mpe

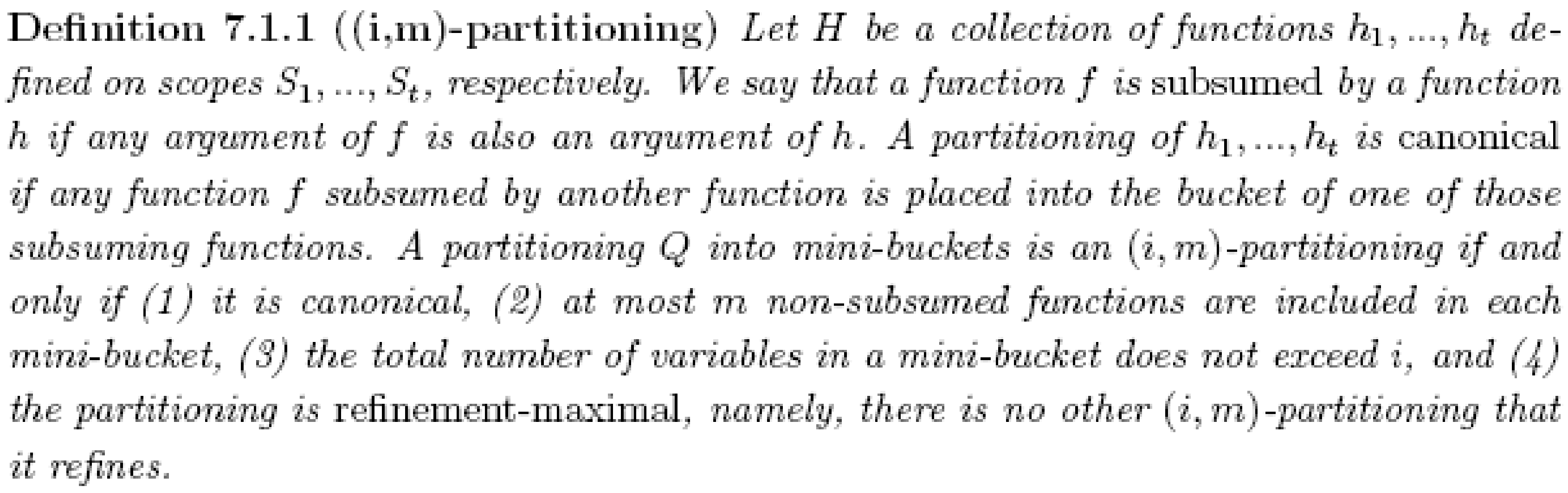


$w^* = 2$



$w^* = 4$

(i,m) patitionings



Definition 7.1.1 ((i,m)-partitioning) *Let H be a collection of functions $h_1, ..., h_t$ defined on scopes $S_1, ..., S_t$, respectively. We say that a function f* is subsumed *by a function h if any argument of f is also an argument of h. A partitioning of $h_1, ..., h_t$ is* canonical *if any function f subsumed by another function is placed into the bucket of one of those subsuming functions. A partitioning Q into mini-buckets is an (i, m)-partitioning if and only if (1) it is canonical, (2) at most m non-subsumed functions are included in each mini-bucket, (3) the total number of variables in a mini-bucket does not exceed i, and (4) the partitioning is* refinement-maximal, *namely, there is no other (i, m)-partitioning that it refines.*

MBE(i,m), (MBE(i) , approx-mpe)

- Input: Belief network (P1,…Pn)
- Output: upper and lower bounds
- Initialize: (put functions in buckets)
- Process each bucket from p=n to 1
    - Create (i,m)-mini-buckets
    - Process each mini-bucket
- (For mpe): assign values in ordering d
- Return: mpe-tuple, upper and lower bounds

```
Algorithm mbe-mpe(i,m)
Input: A belief network $BN = (G, P)$, an ordering $o$, evidence $\bar{e}$.
Output: An upper bound $U$ and a lower bound $L$ on the $MPE = \max_{\bar{x}} P(\bar{x}, \bar{e})$,
and a suboptimal solution $\bar{x}^a$ that provides $L = P(\bar{x}^a)$.
1. Initialize: Partition $P = \{P_1, ..., P_n\}$ into buckets $bucket_1, \ldots, bucket_n$,
   where $bucket_p$ contains all CPTs $h_1, h_2, ..., h_t$ whose highest-index variable is $X_p$.
2. Backward: for $p = n$ to 2 do
   • If $X_p$ is observed ($X_p = a$), assign $X_p = a$ in each $h_j$ and put the result
     in its highest-variable bucket (put constants in $bucket_1$).
   • Else  for $h_1, h_2, ..., h_t$ in $bucket_p$ do
     Generate an $(i, m)$-mini-bucket-partitioning, $Q' = \{Q_1, ..., Q_r\}$.
     for each $Q_l \in Q'$ containing $h_{l_1}, ...h_{l_t}$, do
         compute $h^l = max_{X_p} \Pi_{j=1}^{t} h_{l_j}$ and place it in the bucket of the highest-index
         variable in $U_l \leftarrow \bigcup_{j=1}^{t} S_{l_j} - \{X_p\}$, where $S_{l_j}$ is the scope of $h_{l_j}$
         (put constants in $bucket_1$).
3. Forward: for $p = 1$ to $n$, given $x_1^a, ..., x_{p-1}^a$, do
   assign a value $x_p^a$ to $X_p$ that maximizes the product of all functions in $bucket_p$.
4. Return the assignment $\bar{x}^a = (x_1^a, ..., x_n^a)$, a lower bound $L = P(\bar{x}^a)$, and
   an upper bound $U = max_{x_1} \prod_{h_j \in bucket_1} h^j$ on the $MPE = \max_{\bar{x}} P(\bar{x}, \bar{e})$.
```

Theorem 7.1.3 (mbe-mpe properties) *Algorithm* mbe-mpe*(i, m) computes an upper bound on the MPE. Its time and space complexity is $O(n \cdot exp(i))$ where $i \leq n$.*

Partitioning refinements

Clearly, as the mini-buckets get smaller, both complexity and accuracy decrease.

Definition 7.1.4 *Given two partitionings Q' and Q'' over the same set of elements, Q' is a refinement of Q'' if and only if for every set $A \in Q'$ there exists a set $B \in Q''$ such that $A \subseteq B$.*

It is easy to see that:

Proposition 7.1.5 *If Q'' is a refinement of Q' in bucket$_p$, then $h^p \leq g^p_{Q'} \leq g^p_{Q''}$.*

Remember that *mbe-mpe* computes the bounds on $MPE = \max_{\bar{x}} P(\bar{x}, \bar{e})$, rather than on $M = \max_{\bar{x}} P(\bar{x}|\bar{e}) = MPE/P(\bar{e})$. Thus

$$\frac{L}{P(\bar{e})} \leq M \leq \frac{U}{P(\bar{e})}$$

Properties of approx-mpe(i)

- **Complexity:** *O(exp(i))* time and *O(exp(i))* space.
- **Accuracy:** determined by upper/lower (U/L) bound.
- As ***i*** increases, both accuracy and complexity increase.
- Possible use of mini-bucket approximations:
  - As anytime algorithms (Dechter and Rish, 1997)
  - As heuristics in best-first search (Kask and Dechter, 1999)

Anytime Approximation

```
anytime-mpe($\varepsilon$)

Initialize: $i = i_0$
While time and space resources are available
        $i \leftarrow i + i_{step}$
        $U \leftarrow$ upper bound computed by approx-mpe(i)
        $L \leftarrow$ lower bound computed by approx-mpe(i)
        if $U = L$, return exact optimal solution (certificate of optimality)
        keep the best solution found so far
        if $1 \le \frac{U}{L} \le 1 + \varepsilon$, return solution
end
return the largest $L$ and the smallest $U$
```

Bounded Inference for Belief Updating for probability of evidence

- Idea mini-bucket is the same:

$$\sum_X f(x) \bullet g(x) \le \sum_X f(x) \bullet \sum_X g(x)$$

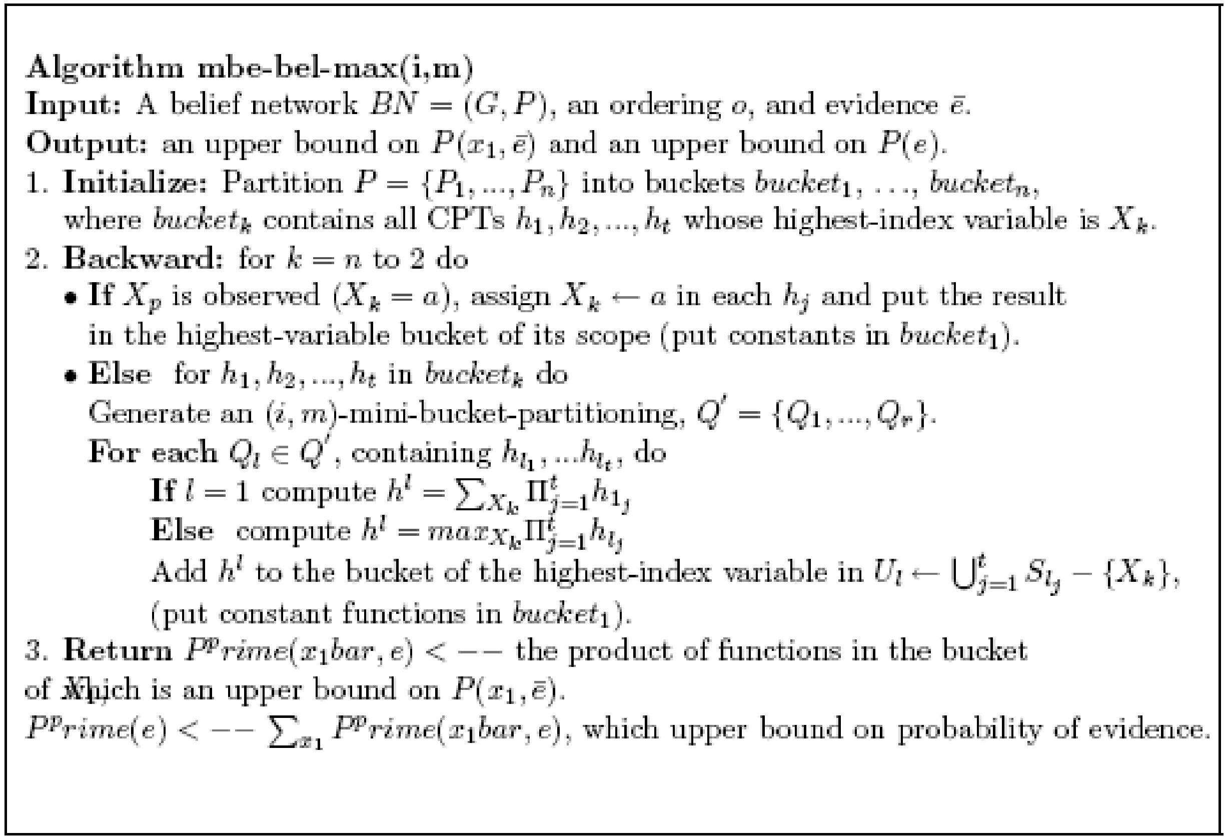
$$\sum_X f(x) \bullet g(x) \le \sum_X f(x) \bullet \max_X g(X)$$

- So we can apply a sum in each mini-bucket, or better, one sum and the rest max, or min (for lower-bound)

- MBE-bel-max(i,m), MBE-bel-min(i,m) generating upper and lower-bound on beliefs approximates BE-bel

- MBE-map(i,m): max buckets will be maximized, sum buckets will be sum-max. Approximates BE-map.

Algorithm mbe-bel-max(i,m)



```
Algorithm mbe-bel-max(i,m)
Input: A belief network $BN = (G, P)$, an ordering $o$, and evidence $\bar{e}$.
Output: an upper bound on $P(x_1, \bar{e})$ and an upper bound on $P(e)$.
1. Initialize: Partition $P = \{P_1, ..., P_n\}$ into buckets $bucket_1, \ldots, bucket_n$,
   where $bucket_k$ contains all CPTs $h_1, h_2, ..., h_t$ whose highest-index variable is $X_k$.
2. Backward: for $k = n$ to 2 do
   • If $X_p$ is observed ($X_k = a$), assign $X_k \leftarrow a$ in each $h_j$ and put the result
     in the highest-variable bucket of its scope (put constants in $bucket_1$).
   • Else  for $h_1, h_2, ..., h_t$ in $bucket_k$ do
     Generate an $(i, m)$-mini-bucket-partitioning, $Q' = \{Q_1, ..., Q_r\}$.
     For each $Q_l \in Q'$, containing $h_{l_1}, ...h_{l_t}$, do
         If $l = 1$ compute $h^l = \sum_{X_k} \Pi_{j=1}^{t} h_{1_j}$
         Else  compute $h^l = max_{X_k} \Pi_{j=1}^{t} h_{l_j}$
         Add $h^l$ to the bucket of the highest-index variable in $U_l \leftarrow \bigcup_{j=1}^{t} S_{l_j} - \{X_k\}$,
         (put constant functions in $bucket_1$).
3. Return $P^prime(x_1bar, e) < --$ the product of functions in the bucket
of $X_1$which is an upper bound on $P(x_1, \bar{e})$.
$P^prime(e) < -- \sum_{x_1} P^prime(x_1bar, e)$, which upper bound on probability of evidence.
```

Figure 7.5: Algorithm *mbe-bel-max(i,m)*.

Empirical Evaluation
(Dechter and Rish, 1997; Rish thesis, 1999)

- Randomly generated networks
  - Uniform random probabilities
  - Random noisy-OR
- CPCS networks
- Probabilistic decoding

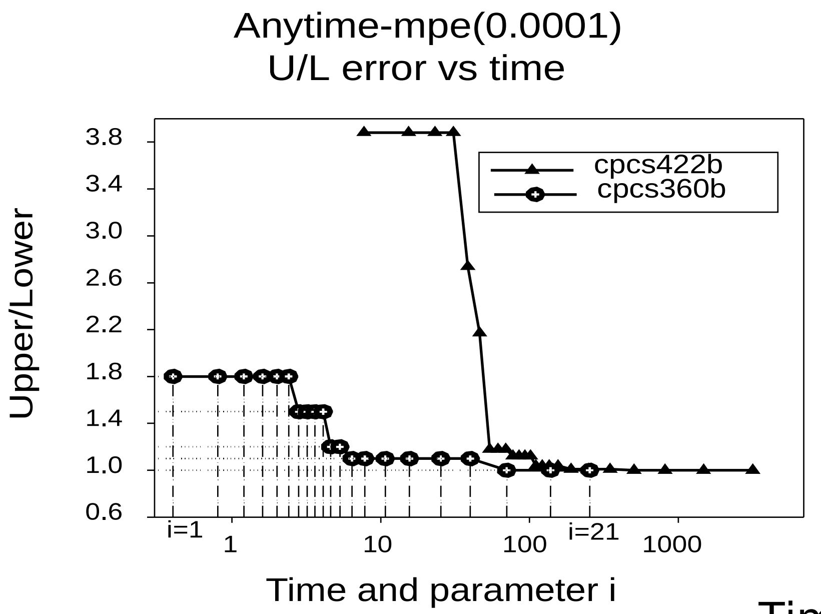
Comparing MBE-mpe and anytime-mpe
versus BE-mpe

Methodology for Empirical Evaluation (for mpe)

- U/L –accuracy
- Better (U/mpe) or mpe/L
- Benchmarks: Random networks
  - Given n,e,v generate a random DAG
  - For xi and parents generate table from uniform [0,1], or noisy-or
- Create k instances. For each, generate random evidence, likely evidence
- Measure averages

CPCS networks – medical diagnosis (noisy-OR model)

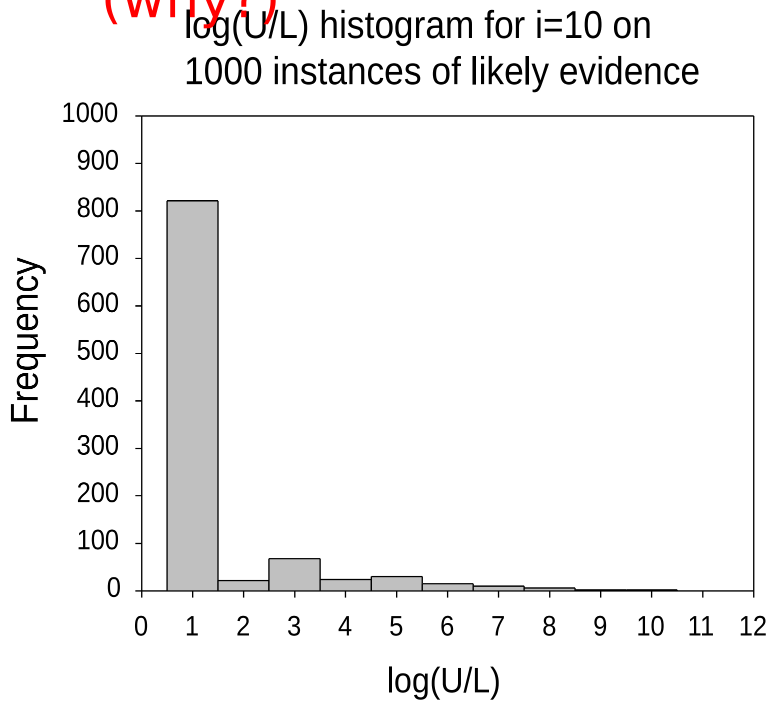
Test case: no evidence

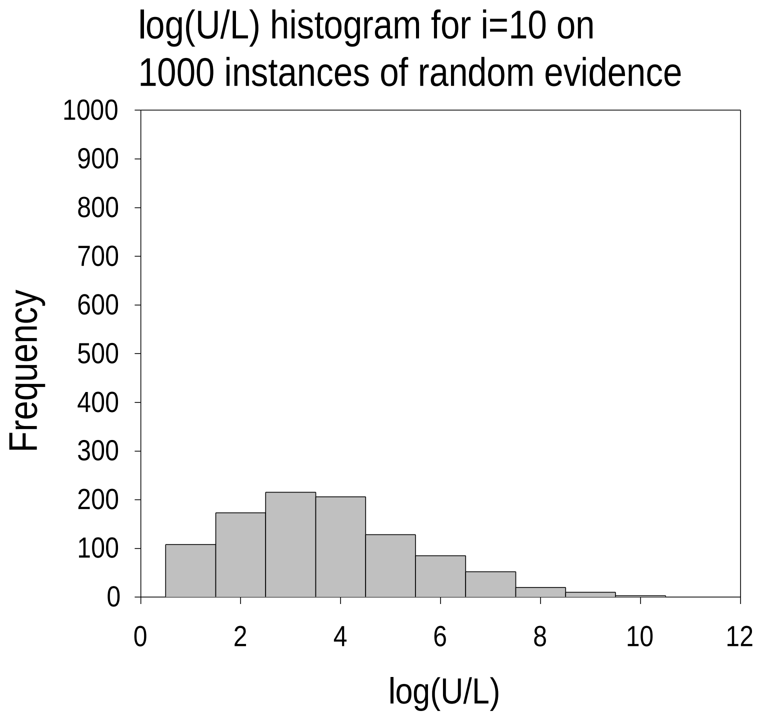


	Time (sec)	
Algorithm	cpcs360	cpcs422
elim-mpe	115.8	1697.6
anytime-mpe(ϵ), $\epsilon = 10^{-4}$	70.3	505.2
anytime-mpe(ϵ), $\epsilon = 10^{-1}$	70.3	110.5

The effect of evidence

More likely evidence=>higher MPE => higher accuracy (why?)

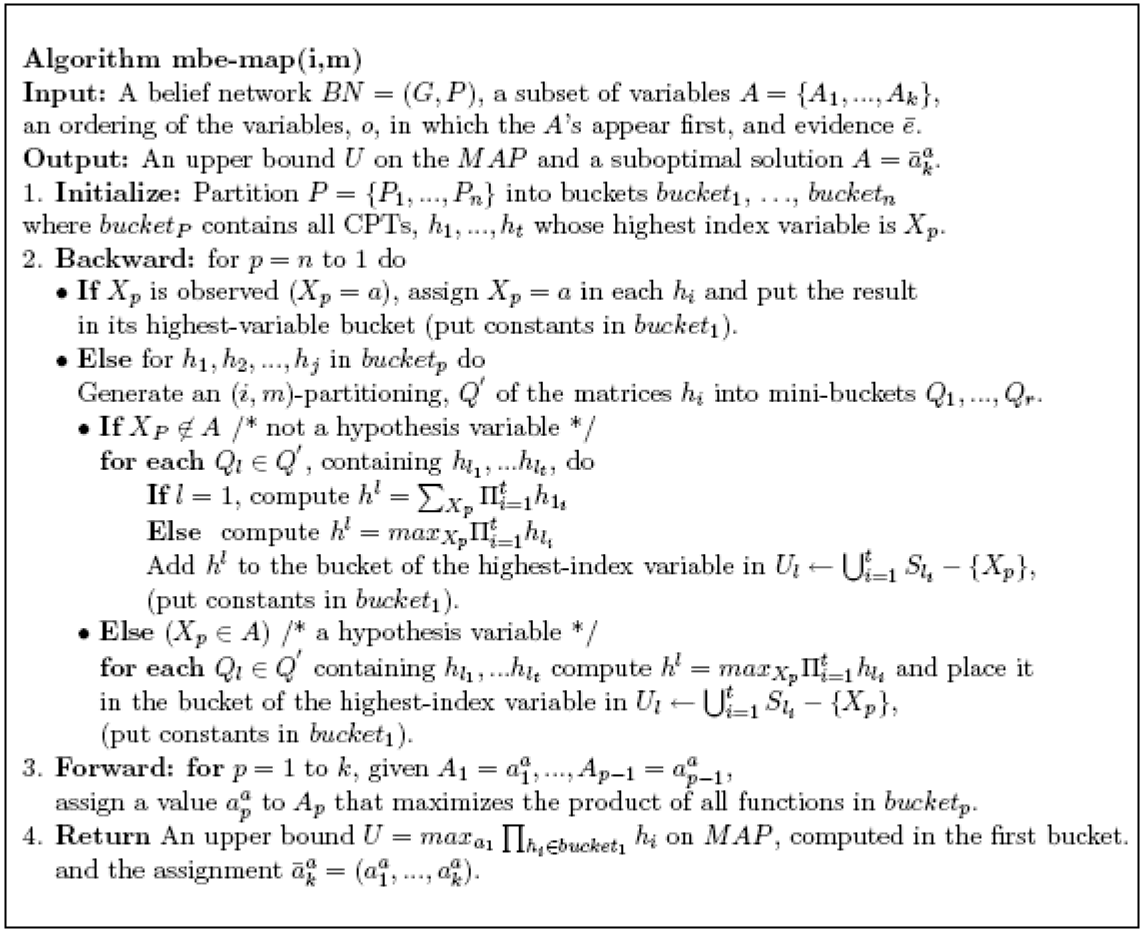




Likely evidence versus random (unlikely) evidence

MBE-map

Process max buckets
With max mini-buckets
And sum buckets with su
Mini-bucket and max
mini-buckets

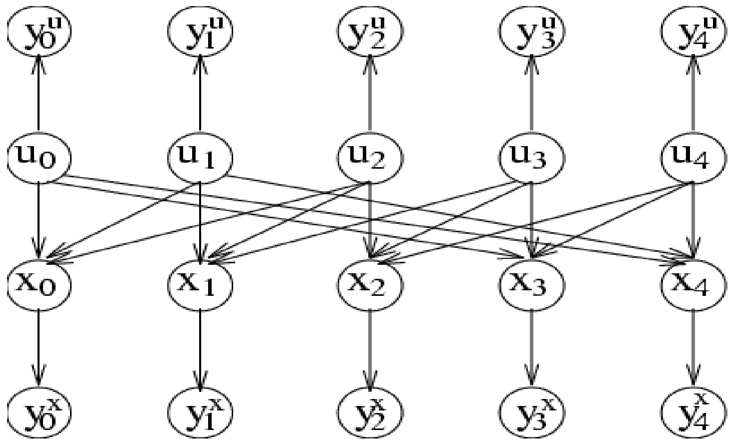


```
Algorithm mbe-map(i,m)
Input: A belief network $BN = (G, P)$, a subset of variables $A = \{A_1, ..., A_k\}$,
an ordering of the variables, $o$, in which the $A$'s appear first, and evidence $\bar{e}$.
Output: An upper bound $U$ on the $MAP$ and a suboptimal solution $A = \bar{a}^a_k$.
1. Initialize: Partition $P = \{P_1, ..., P_n\}$ into buckets $bucket_1, \ldots, bucket_n$
where $bucket_P$ contains all CPTs, $h_1, ..., h_t$ whose highest index variable is $X_p$.
2. Backward: for $p = n$ to 1 do
   • If $X_p$ is observed ($X_p = a$), assign $X_p = a$ in each $h_i$ and put the result
     in its highest-variable bucket (put constants in $bucket_1$).
   • Else for $h_1, h_2, ..., h_j$ in $bucket_p$ do
     Generate an $(i,m)$-partitioning, $Q'$ of the matrices $h_i$ into mini-buckets $Q_1, ..., Q_r$.
     • If $X_P \notin A$ /* not a hypothesis variable */
       for each $Q_l \in Q'$, containing $h_{l_1}, ...h_{l_t}$, do
           If $l = 1$, compute $h^l = \sum_{X_p} \Pi_{i=1}^t h_{1_i}$
           Else  compute $h^l = max_{X_p} \Pi_{i=1}^t h_{l_i}$
           Add $h^l$ to the bucket of the highest-index variable in $U_l \leftarrow \bigcup_{i=1}^t S_{l_i} - \{X_p\}$,
           (put constants in $bucket_1$).
     • Else $(X_p \in A)$ /* a hypothesis variable */
       for each $Q_l \in Q'$ containing $h_{l_1}, ...h_{l_t}$ compute $h^l = max_{X_p} \Pi_{i=1}^t h_{l_i}$ and place it
       in the bucket of the highest-index variable in $U_l \leftarrow \bigcup_{i=1}^t S_{l_i} - \{X_p\}$,
       (put constants in $bucket_1$).
3. Forward: for $p = 1$ to $k$, given $A_1 = a^a_1, ..., A_{p-1} = a^a_{p-1}$,
   assign a value $a^a_p$ to $A_p$ that maximizes the product of all functions in $bucket_p$.
4. Return An upper bound $U = max_{a_1} \prod_{h_i \in bucket_1} h_i$ on $MAP$, computed in the first bucket.
   and the assignment $\bar{a}^a_k = (a^a_1, ..., a^a_k)$.
```

Figure 7.6: Algorithm *mbe-map(i,m)*.

Probabilistic decoding

Error-correcting linear block code



State-of-the-art:
approximate algorithm – iterative belief propagation (IBP)
(Pearl’s poly-tree algorithm applied to loopy networks)

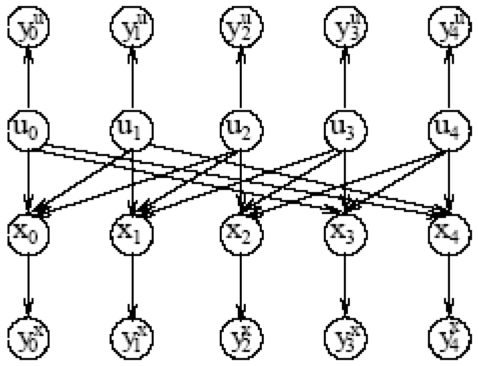


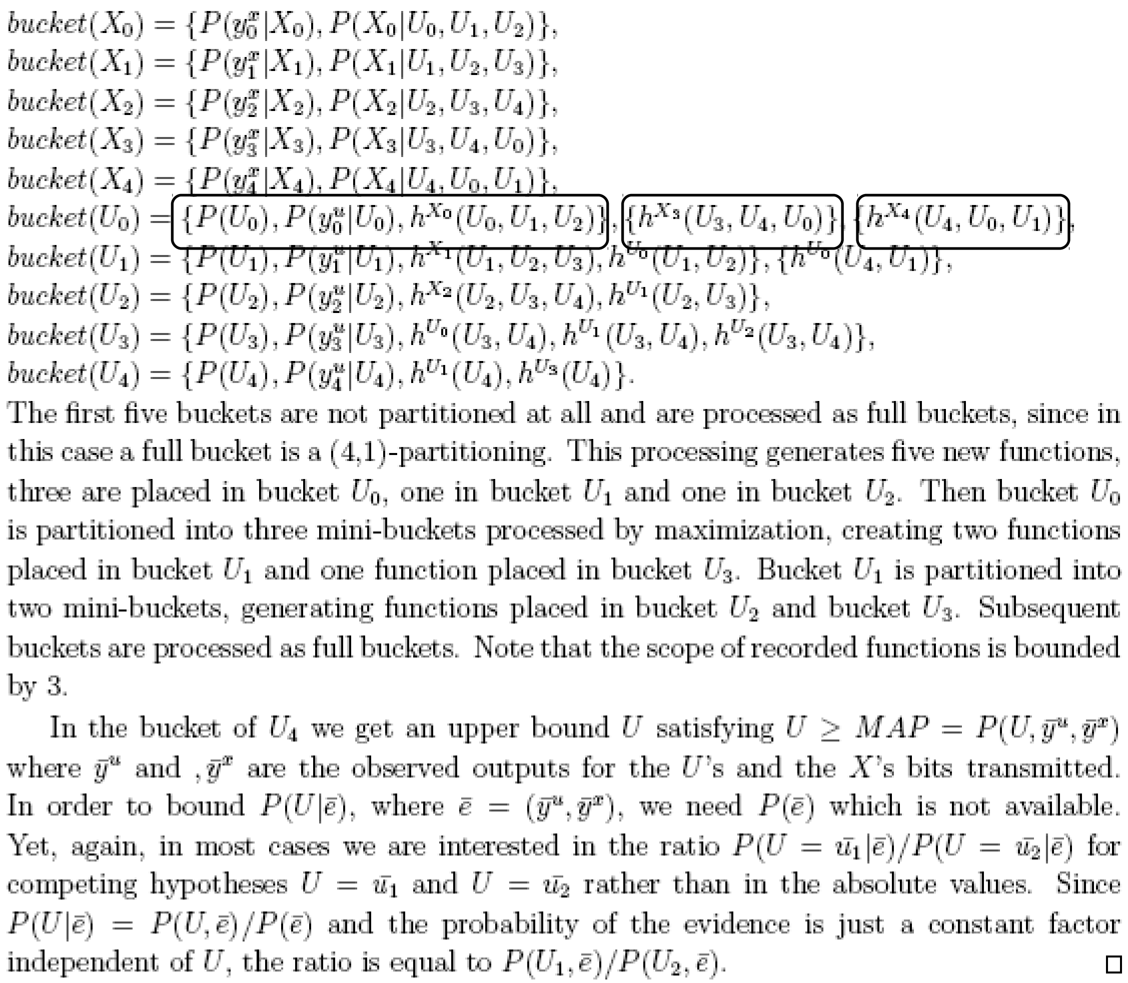
Figure 7.7: Belief network for a linear block code.

Example 7.3.1 We will next demonstrate the mini-bucket approximation for MAP on an example of *probabilistic decoding* (see Chapter 2) Consider a belief network which describes the decoding of a *linear block code*, shown in Figure 7.7. In this network, U_i are *information bits* and X_j are *code bits*, which are functionally dependent on U_i. The vector (U, X), called the channel input, is transmitted through a noisy channel which adds Gaussian noise and results in the channel output vector $Y = (Y^u, Y^x)$. The decoding task is to assess the most likely values for the U's given the observed values $Y = (\bar{y}^u, \bar{y}^x)$, which is the MAP task where U is the set of hypothesis variables, and $Y = (\bar{y}^u, \bar{y}^x)$ is the evidence. After processing the observed buckets we get the following bucket configuration (lower case y's are observed values):

$$\left.\begin{aligned}
bucket(X_0) &= P(y_0^x|X_0), P(X_0|U_0, U_1, U_2),\\
bucket(X_1) &= P(y_1^x|X_1), P(X_1|U_1, U_2, U_3),\\
bucket(X_2) &= P(y_2^x|X_2), P(X_2|U_2, U_3, U_4),\\
bucket(X_3) &= P(y_3^x|X_3), P(X_3|U_3, U_4, U_0),\\
bucket(X_4) &= P(y_4^x|X_4), P(X_4|U_4, U_0, U_1),\\
bucket(U_0) &= P(U_0), P(y_0^u|U_0),\\
bucket(U_1) &= P(U_1), P(y_1^u|U_1),\\
bucket(U_2) &= P(U_2), P(y_2^u|U_2),\\
bucket(U_3) &= P(U_3), P(y_3^u|U_3),\\
bucket(U_4) &= P(U_4), P(y_4^u|U_4).
\end{aligned}\right\}$$

Initial partitioning

Processing by *mbe-map(4,1)* of the first top five buckets by summation and the rest by maximization, results in the following mini-bucket partitionings and function generation:

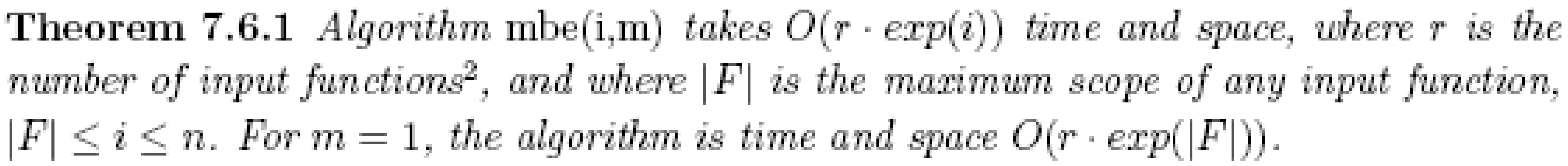


$bucket(X_0) = \{P(y_0^x|X_0), P(X_0|U_0,U_1,U_2)\},$
$bucket(X_1) = \{P(y_1^x|X_1), P(X_1|U_1,U_2,U_3)\},$
$bucket(X_2) = \{P(y_2^x|X_2), P(X_2|U_2,U_3,U_4)\},$
$bucket(X_3) = \{P(y_3^x|X_3), P(X_3|U_3,U_4,U_0)\},$
$bucket(X_4) = \{P(y_4^x|X_4), P(X_4|U_4,U_0,U_1)\},$
$bucket(U_0) = \boxed{\{P(U_0), P(y_0^u|U_0), h^{X_0}(U_0,U_1,U_2)\}}, \boxed{\{h^{X_3}(U_3,U_4,U_0)\}}, \boxed{\{h^{X_4}(U_4,U_0,U_1)\}},$
$bucket(U_1) = \{P(U_1), P(y_1^u|U_1), h^{X_1}(U_1,U_2,U_3), h^{U_0}(U_1,U_2)\}, \{h^{U_0}(U_4,U_1)\},$
$bucket(U_2) = \{P(U_2), P(y_2^u|U_2), h^{X_2}(U_2,U_3,U_4), h^{U_1}(U_2,U_3)\},$
$bucket(U_3) = \{P(U_3), P(y_3^u|U_3), h^{U_0}(U_3,U_4), h^{U_1}(U_3,U_4), h^{U_2}(U_3,U_4)\},$
$bucket(U_4) = \{P(U_4), P(y_4^u|U_4), h^{U_1}(U_4), h^{U_3}(U_4)\}.$

The first five buckets are not partitioned at all and are processed as full buckets, since in this case a full bucket is a (4,1)-partitioning. This processing generates five new functions, three are placed in bucket U_0, one in bucket U_1 and one in bucket U_2. Then bucket U_0 is partitioned into three mini-buckets processed by maximization, creating two functions placed in bucket U_1 and one function placed in bucket U_3. Bucket U_1 is partitioned into two mini-buckets, generating functions placed in bucket U_2 and bucket U_3. Subsequent buckets are processed as full buckets. Note that the scope of recorded functions is bounded by 3.

In the bucket of U_4 we get an upper bound U satisfying $U \geq MAP = P(U, \bar{y}^u, \bar{y}^x)$ where $\bar{y}^u$ and $, \bar{y}^x$ are the observed outputs for the U's and the X's bits transmitted. In order to bound $P(U|\bar{e})$, where $\bar{e} = (\bar{y}^u, \bar{y}^x)$, we need $P(\bar{e})$ which is not available. Yet, again, in most cases we are interested in the ratio $P(U = \bar{u}_1|\bar{e})/P(U = \bar{u}_2|\bar{e})$ for competing hypotheses $U = \bar{u}_1$ and $U = \bar{u}_2$ rather than in the absolute values. Since $P(U|\bar{e}) = P(U,\bar{e})/P(\bar{e})$ and the probability of the evidence is just a constant factor independent of U, the ratio is equal to $P(U_1,\bar{e})/P(U_2,\bar{e})$. □

Complexity and tractability of MBE(i,m)

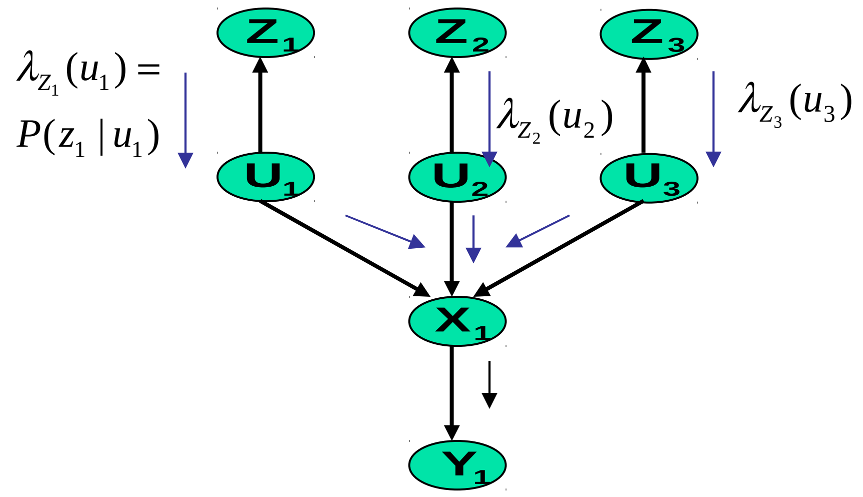


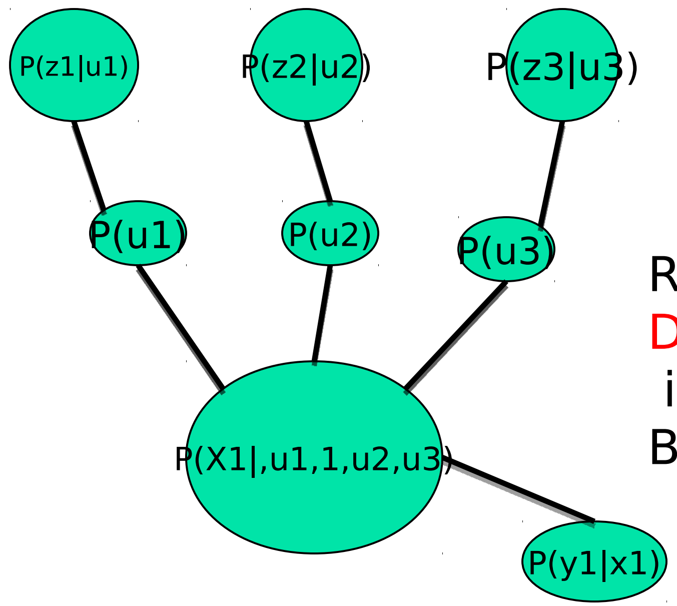
Theorem 7.6.1 *Algorithm* mbe(i,m) *takes* $O(r \cdot exp(i))$ *time and space, where* r *is the number of input functions*[2], *and where* $|F|$ *is the maximum scope of any input function,* $|F| \leq i \leq n$. *For* $m = 1$, *the algorithm is time and space* $O(r \cdot exp(|F|))$.

Belief propagation is easy on polytree: Pearl's Belief Propagation

A polytree: a tree with
Larger families

A polytree decomposition

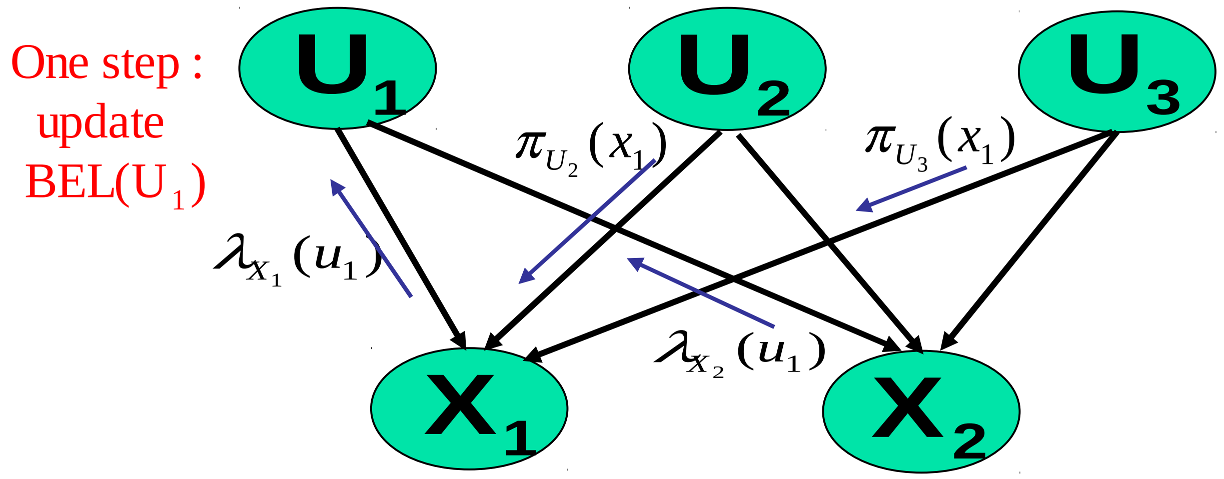




Running CTE = running Pearl's BP over the dual gra
Dual-graph: nodes are cpts, arcs connect non-empty
intersections.
BP is Time and space linear

Iterative Belief Proapagation

- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks

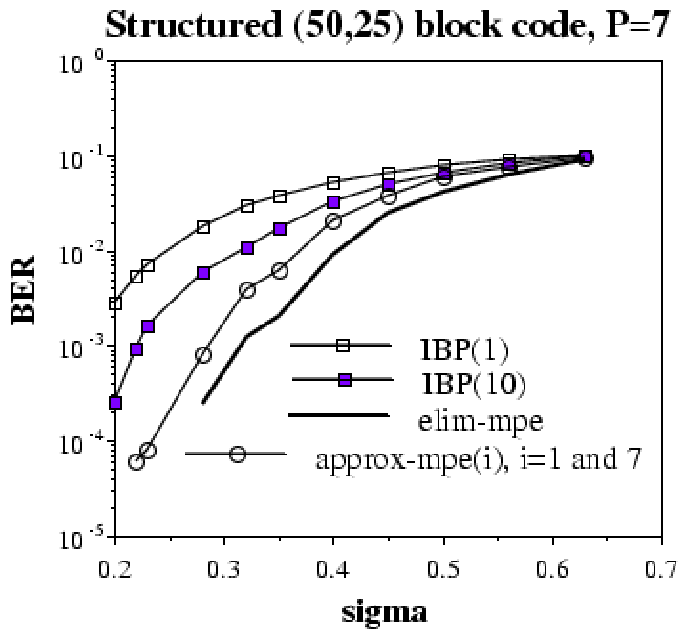


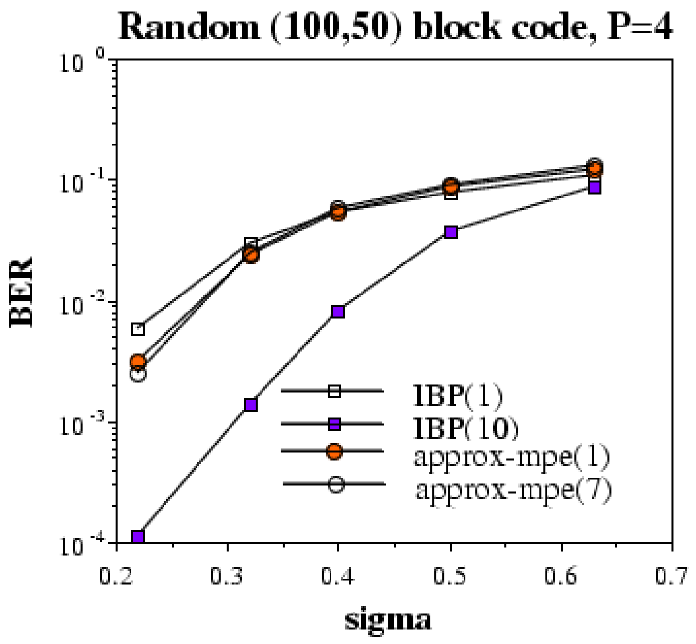
- No guarantees for convergence
- Works well for many coding networks

MBE-mpe vs. IBP

approx - mpe is better on low - w * codes
IBP is better on randomly generated (high - w*) codes

Bit error rate (BER) as a function of noise (sigma):





Mini-buckets: summary

- Mini-buckets – local inference approximation
- Idea: bound size of recorded functions
- Approx-mpe(i) - mini-bucket algorithm for MPE
  - Better results for noisy-OR than for random problems
  - Accuracy increases with decreasing noise in coding
  - Accuracy increases for likely evidence
  - Sparser graphs -> higher accuracy
  - Coding networks: approx-mpe outperfroms IBP on low-induced width codes

Cluster Tree Elimination - properties

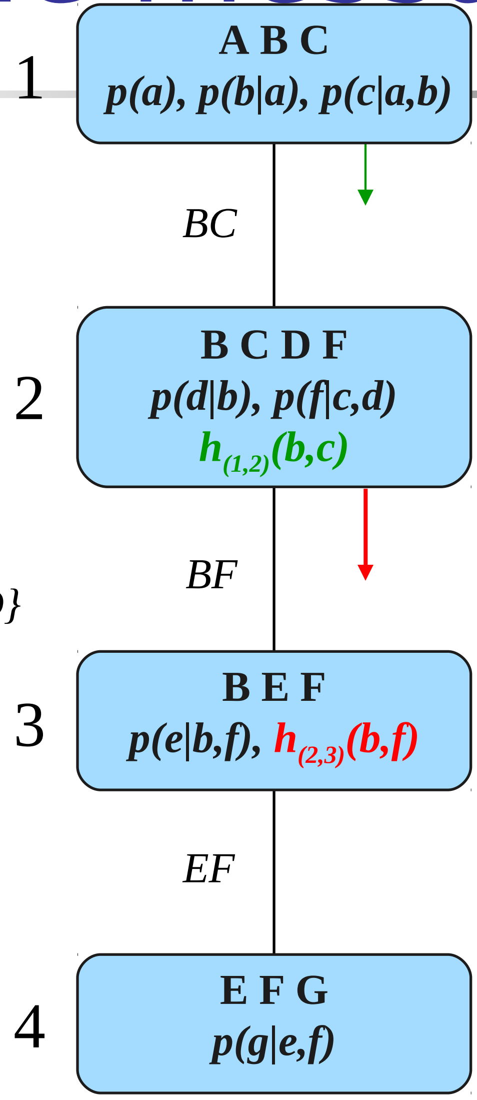
- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability of a single variable and the evidence.

- Time complexity: $O\ (\ deg \times (n+N) \times d^{\ w*+1}\ )$

- Space complexity: $O\ (\ N \times d^{\ sep})$

where *deg* = the maximum degree of a node
n = number of variables (= number of CPTs)
N = number of nodes in the tree decomposition
d = the maximum domain size of a variable
*w** = the induced width
sep = the separator size

Cluster Tree Elimination - the messages



$$h_{(1,2)}(b,c) = \sum_{a} p(a) \cdot p(b|a) \cdot p(c|a,b)$$

sep(2,3)={B,F}

elim(2,3)={C,D}

$$h_{(2,3)}(b,f) = \sum_{c,d} p(d|b) \cdot p(f|c,d) \cdot h_{(1,2)}(b,c)$$

Mini-Clustering for belief updating

- Motivation:
  - Time and space complexity of Cluster Tree Elimination depend on the induced width *w** of the problem
  - When the induced width *w** is big, CTE algorithm becomes infeasible

- The basic idea:
  - Try to reduce the size of the cluster (the exponent);
    partition each cluster into mini-clusters with less variables
  - Accuracy parameter *i* = maximum number of variables in a mini-cluster
  - The idea was explored for variable elimination (Mini-Bucket)

Idea of Mini-Clustering

Split a cluster into mini-clusters => bound complexity

$$\begin{tikzcd}
 & cluster(u) = \{h_1,\ldots,h_r,h_{r+1},\ldots,h_n\} \arrow[d] \arrow[ddl] \arrow[ddr] & \\
 & h = \sum_{elim}\prod_{i=1}^{n} h_i & \\
\{h_1,\ldots,h_r\} \arrow[dr] & & \{h_{r+1},\ldots,h_n\} \arrow[dl] \\
 & g = \left(\sum_{elim}\prod_{i=1}^{r} h_i\right)\cdot\left(\sum_{elim}\prod_{i=r+1}^{n} h_i\right) &
\end{tikzcd}$$

$$h \le g$$

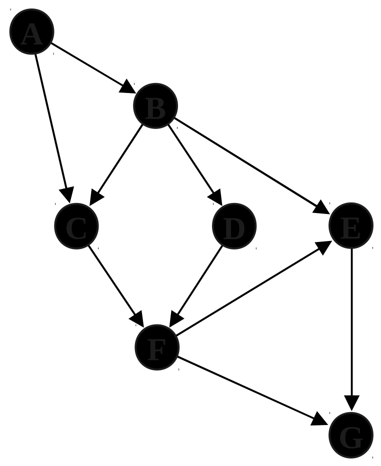
Exponential complexity decrease: $O(e^n) \rightarrow O(e^r) + O(e^{n-r})$

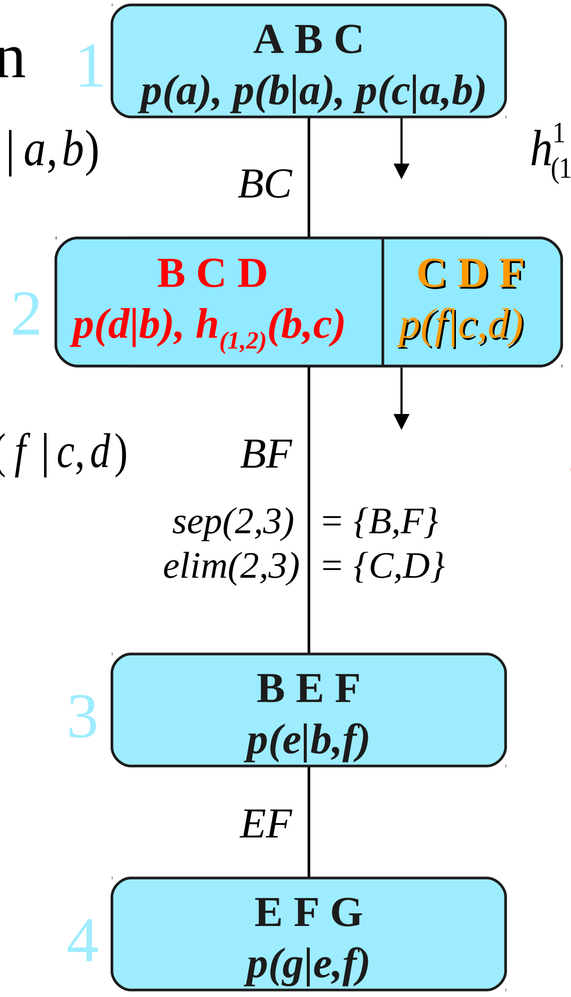
Mini-Clustering - MC

Cluster Tree Elimination

$$h_{(1,2)}(b,c) = \sum_a p(a) \cdot p(b|a) \cdot p(c|a,b)$$

$$h_{(2,3)}(b,f) = \sum_{c,d} p(d|b) \cdot h^1_{(1,2)}(b,c) \cdot p(f|c,d)$$





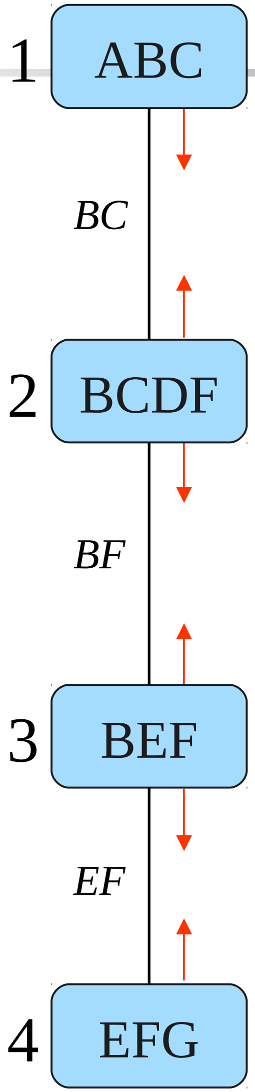
Mini-Clustering, i=3

$$h^1_{(1,2)}(b,c) = \sum_a p(a) \cdot p(b|a) \cdot p(c|a,b)$$

$$h^1_{(2,3)}(b) = \sum_{c,d} p(d|b) \cdot h^1_{(1,2)}(b,c)$$

$$h^2_{(2,3)}(f) = \sum_{c,d} p(f|c,d)$$

Mini-Clustering - example



$H_{(1,2)}$ $\quad h^1_{(1,2)}(b,c) := \sum_a p(a) \cdot p(b|a) \cdot p(c|a,b)$

$h^1_{(2,1)}(b) := \sum_{d,f} p(d|b) \cdot h^1_{(3,2)}(b,f)$

$H_{(2,1)}$ $\quad h^2_{(2,1)}(c) := \max_{d,f} p(f|c,d)$

$h^1_{(2,3)}(b) := \sum_{c,d} p(d|b) \cdot h^1_{(1,2)}(b,c)$

$H_{(2,3)}$

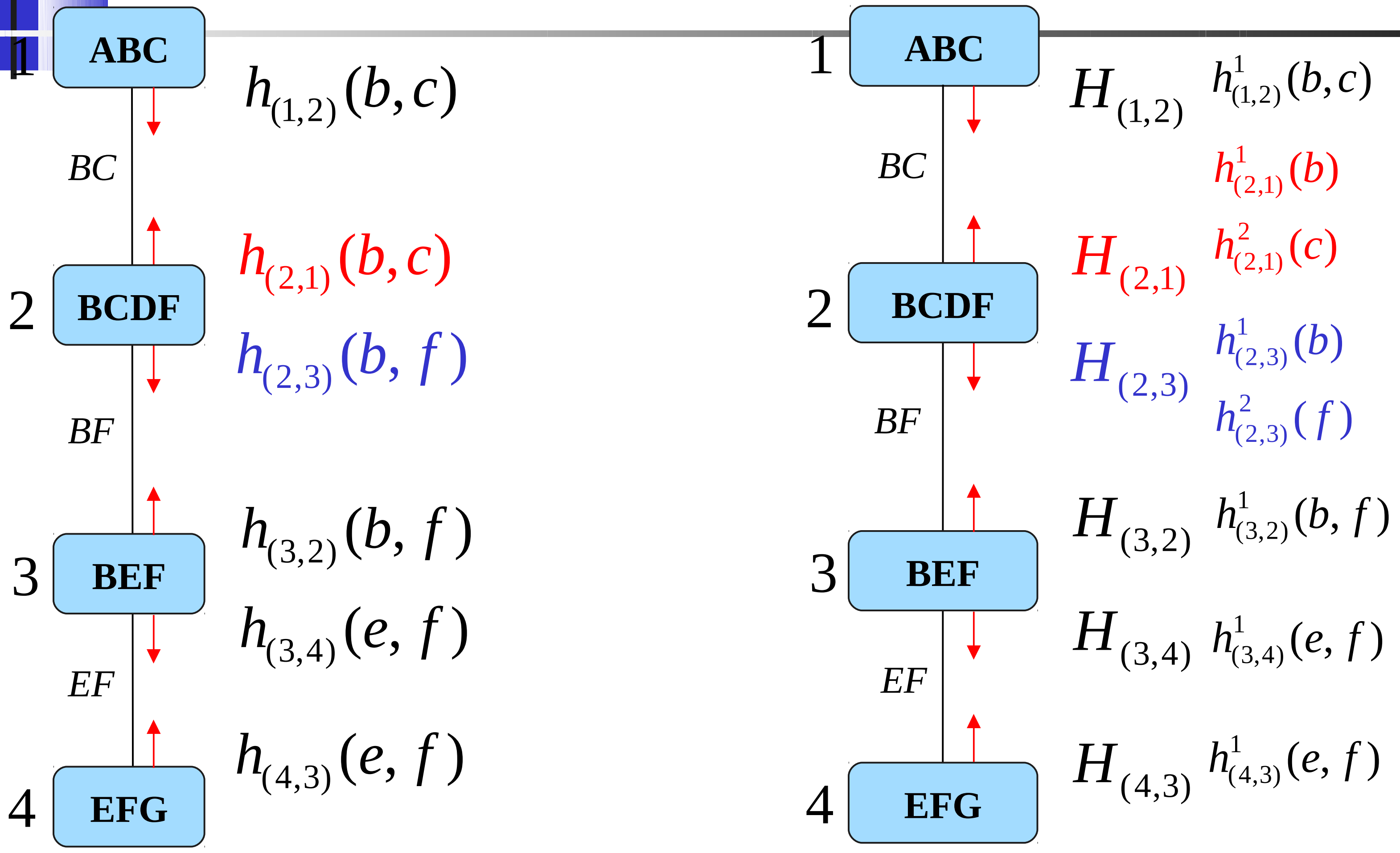
$h^2_{(2,3)}(f) := \max_{c,d} p(f|c,d)$

$H_{(3,2)}$ $\quad h^1_{(3,2)}(b,f) := \sum_e p(e|b,f) \cdot h^1_{(4,3)}(e,f)$

$H_{(3,4)}$ $\quad h^1_{(3,4)}(e,f) := \sum_b p(e|b,f) \cdot h^1_{(2,3)}(b) \cdot h^2_{(2,3)}(f)$

$H_{(4,3)}$ $\quad h^1_{(4,3)}(e,f) := p(G = g_e|e,f)$

Cluster Tree Elimination vs. Mini-Clustering



Semantic of node duplication for mini-clustering

- We can have a different duplication of nodes going up and down. Example: going down.

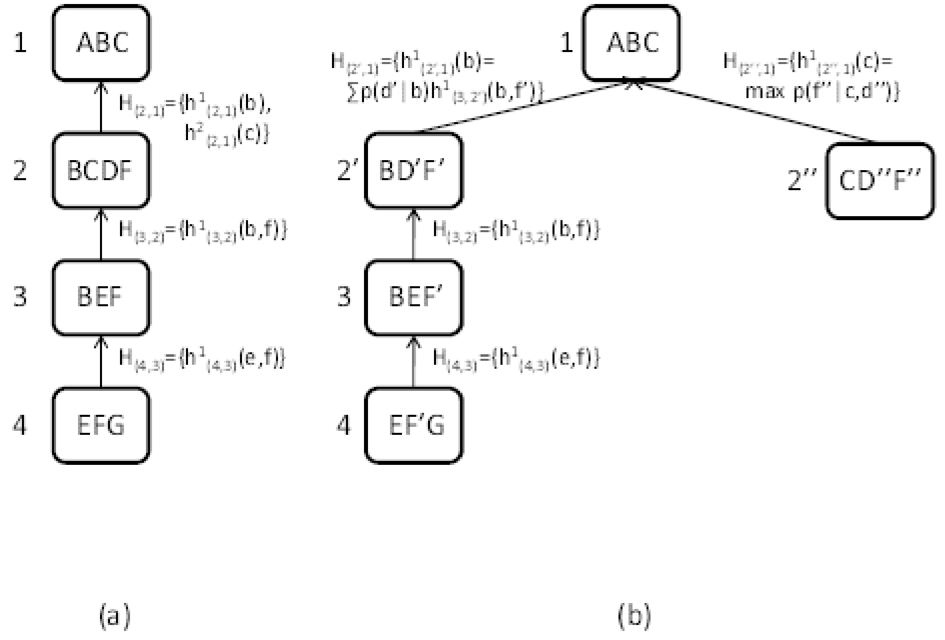
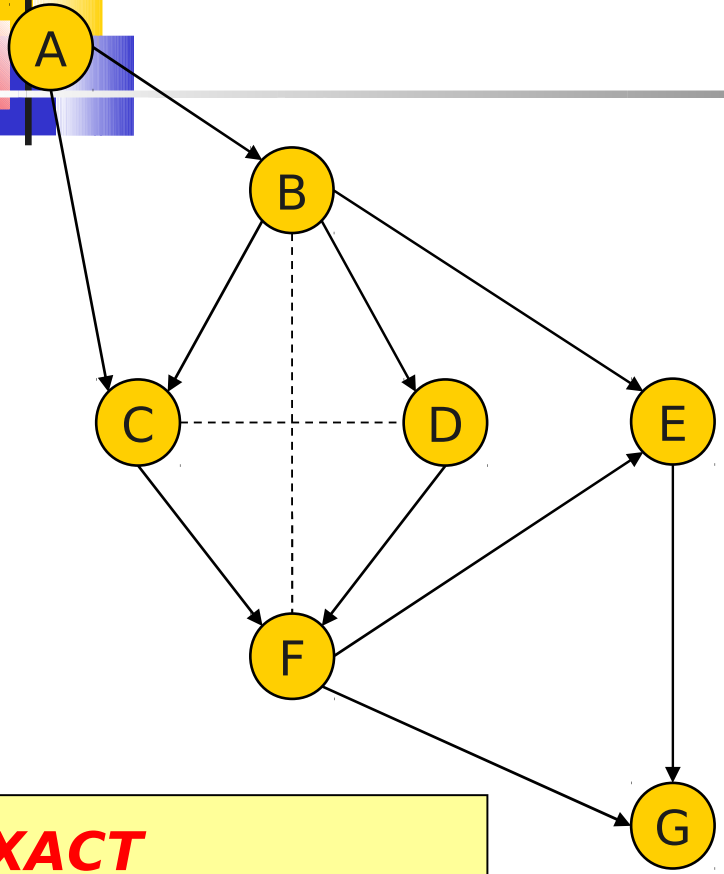
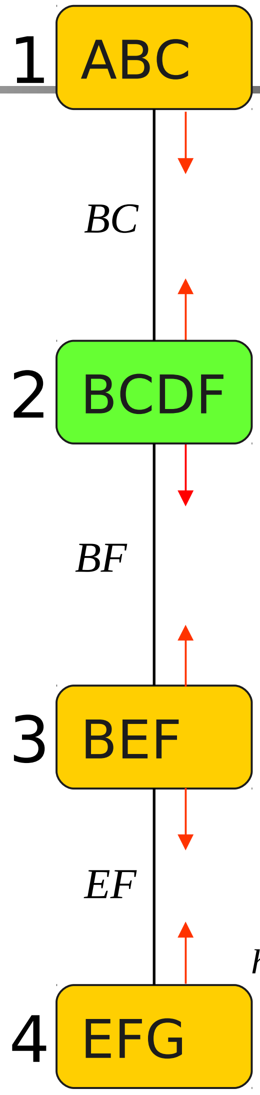

1
ABC
$H_{(2,1)}=\{h^1_{(2,1)}(b), h^2_{(2,1)}(c)\}$
2
BCDF
$H_{(3,2)}=\{h^1_{(3,2)}(b,f)\}$
3
BEF
$H_{(4,3)}=\{h^1_{(4,3)}(e,f)\}$
4
EFG
(a)
1
ABC
$H_{(2',1)}=\{h^1_{(2',1)}(b)=\sum p(d'|b)h^1_{(3,2')}(b,f')\}$
$H_{(2'',1)}=\{h^1_{(2'',1)}(c)=\max p(f''|c,d'')\}$
2'
BD'F'
2''
CD''F''
$H_{(3,2)}=\{h^1_{(3,2)}(b,f)\}$
3
BEF'
$H_{(4,3)}=\{h^1_{(4,3)}(e,f)\}$
4
EF'G
(b)

Figure 1.14: Node duplication semantics of MC: (a) trace of MC-BU(3); (b) trace of CTE-BU.

Join-Tree Clustering





$$h_{(1,2)}(b,c) = \sum_{a} p(a) \cdot p(b|a) \cdot p(c|a,b)$$

$$h_{(2,1)}(b,c) = \sum_{d,f} p(d|b) \cdot p(f|c,d) \cdot h_{(3,2)}(b,f)$$

$$h_{(2,3)}(b,f) = \sum_{c,d} p(d|b) \cdot p(f|c,d) \cdot h_{(1,2)}(b,c)$$

$$h_{(3,2)}(b,f) = \sum_{e} p(e|b,f) \cdot h_{(4,3)}(e,f)$$

$$h_{(3,4)}(e,f) = \sum_{b} p(e|b,f) \cdot h_{(2,3)}(b,f)$$

$$h_{(4,3)}(e,f) = p(G = g_e | e,f)$$

EXACT algorithm

Time and space:
exp(cluster size)= exp(treewidth)

Mini-Clustering

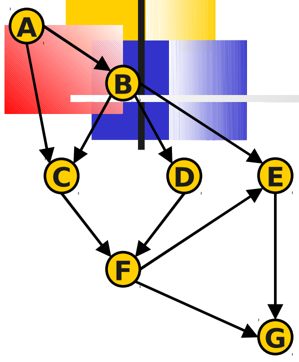
Split a cluster into mini-clusters => bound complexity

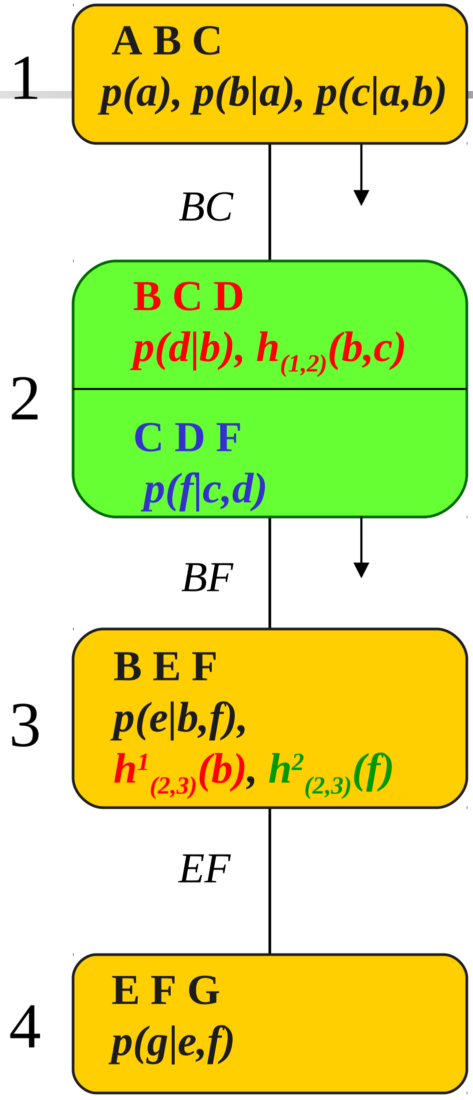
APPROXIMATE algorithm

$$\begin{tikzcd}
 & \{h_1,\ldots,h_r,h_{r+1},\ldots,h_n\} \arrow[dl] \arrow[d] \arrow[dr] & \\
 & \{h_1,\ldots,h_r\} \arrow[d] & \{h_{r+1},\ldots,h_n\} \arrow[d] \\
\sum_{elim}\prod_{i=1}^{n} h_i \quad \leq & \left(\sum_{elim}\prod_{i=1}^{r} h_i\right) \cdot & \left(\sum_{elim}\prod_{i=r+1}^{n} h_i\right)
\end{tikzcd}$$

Exponential complexity decrease $O(e^n) \rightarrow O(e^{\mathrm{var}(r)}) + O(e^{\mathrm{var}(n-r)})$

Mini-Clustering, i-bound=3





$$h^1_{(1,2)}(b,c) = \sum_a p(a)\cdot p(b\,|\,a)\cdot p(c\,|\,a,b)$$

$$h^1_{(2,3)}(b) = \sum_{c,d} p(d\,|\,b)\cdot h^1_{(1,2)}(b,c)$$

$$h^2_{(2,3)}(f) = \max_{c,d} p(f\,|\,c,d)$$

APPROXIMATE algorithm
Time and space:
exp(i-bound)

$\downarrow$

Number of variables in a mini cluster

Mini-Clustering

- **Correctness and completeness**: Algorithm MC(*i*) computes a bound (or an approximation) on the joint probability $P(X_i,e)$ of each variable and each of its values.
- **Time & space** complexity: $O(n \times hw^* \times d^{\,i})$

where $hw^* = max_u \mid \{f \mid f \cap \chi(u) \neq \phi\} \mid$

Lower bounds and mean approximations

We can replace *max* operator by

- *min* => lower bound on the joint
- *mean* => approximation of the joint

Normalization

- MC can compute an (upper) bound on the joint *P(X$_i$,e)*

$$\overline{P}(X_i, e)$$

- Deriving a bound on the conditional *P(X$_i$|e)* is not easy when the exact *P(e)* is not available

- If a lower bound would be available, we could use:

  as an upper bound on the posterior

$$\underline{P}(e)$$

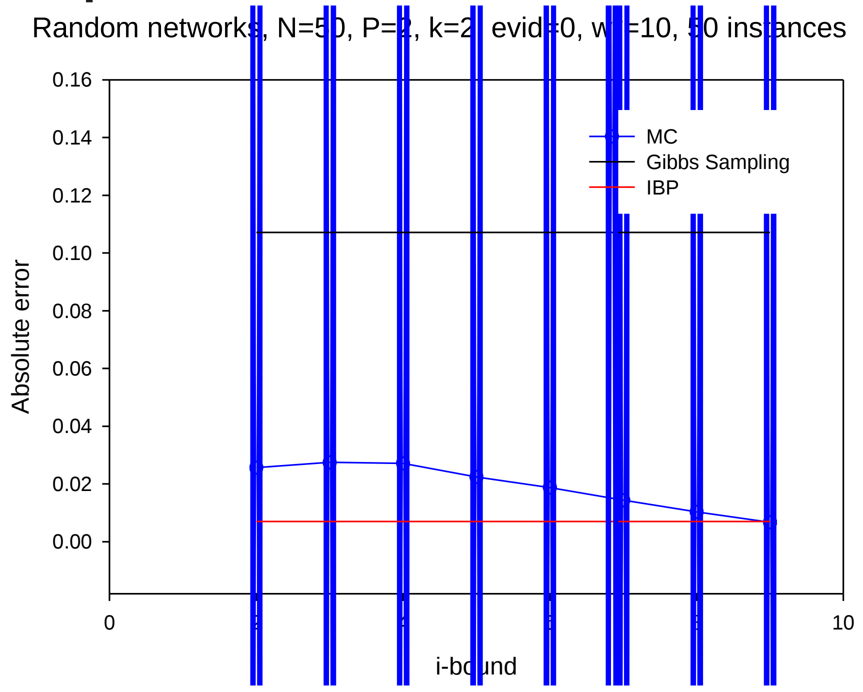
- In our experiments we normalized the results and regarded them as approximations of the posterior *P(X$_i$|e)*

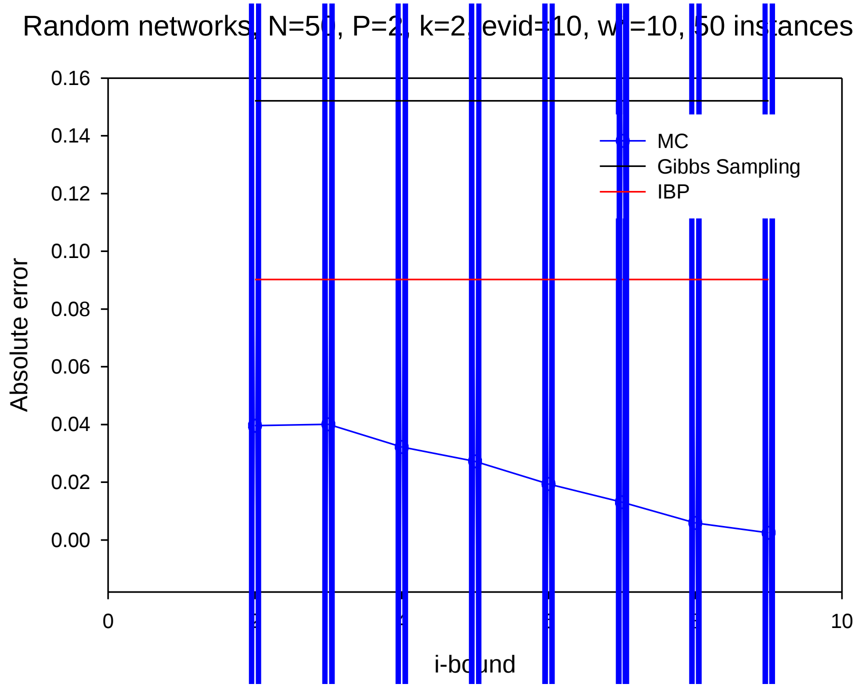
$$\overline{P}(X_i, e) / \underline{P}(e)$$

Experimental results

- Algorithms:
  - Exact
  - IBP
  - Gibbs sampling (GS)
  - MC with normalization (approximate)
- Networks (all variables are binary):
  - Coding networks
  - CPCS 54, 360, 422
  - Grid networks (MxM)
  - Random noisy-OR networks
  - Random networks
- Measures:
  - Normalized Hamming Distance (NHD)
  - BER (Bit Error Rate)
  - Absolute error
  - Relative error
  - Time

Random networks - Absolute error

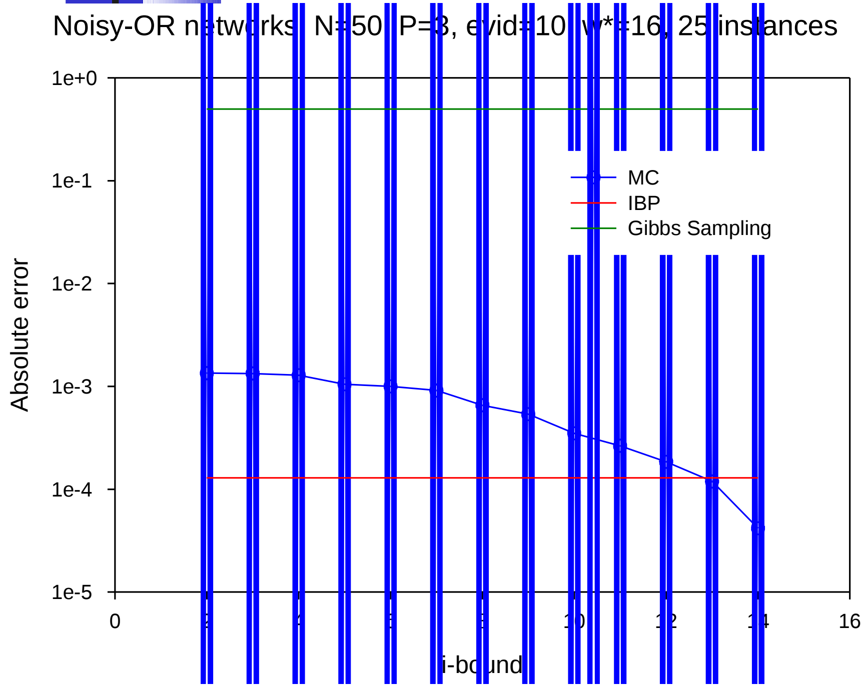


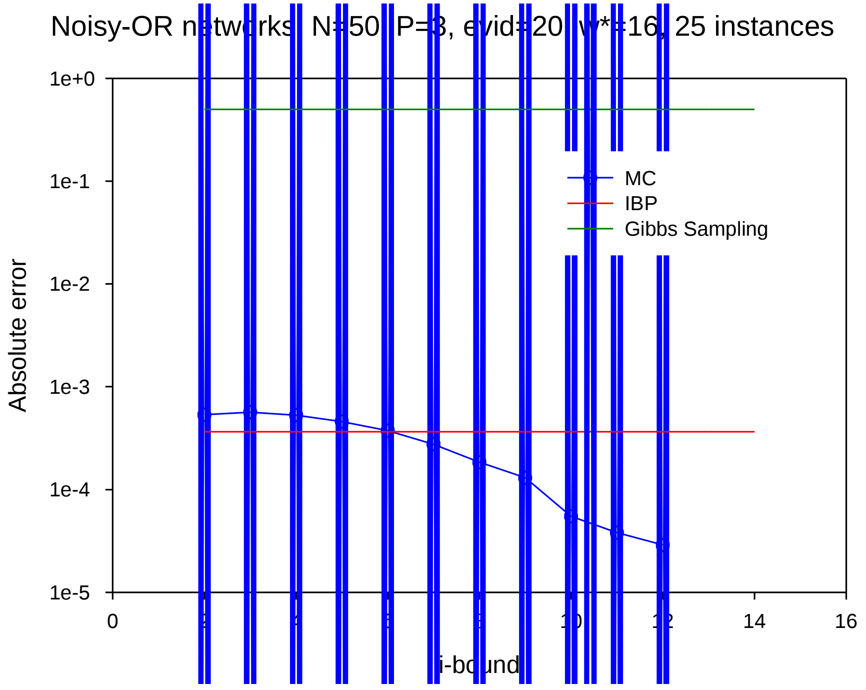


evidence=0

evidence=10

Noisy-OR networks - Absolute error

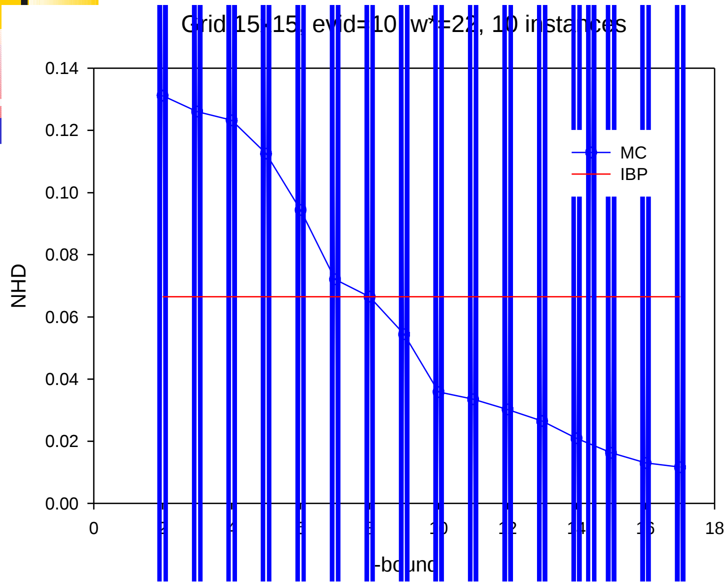


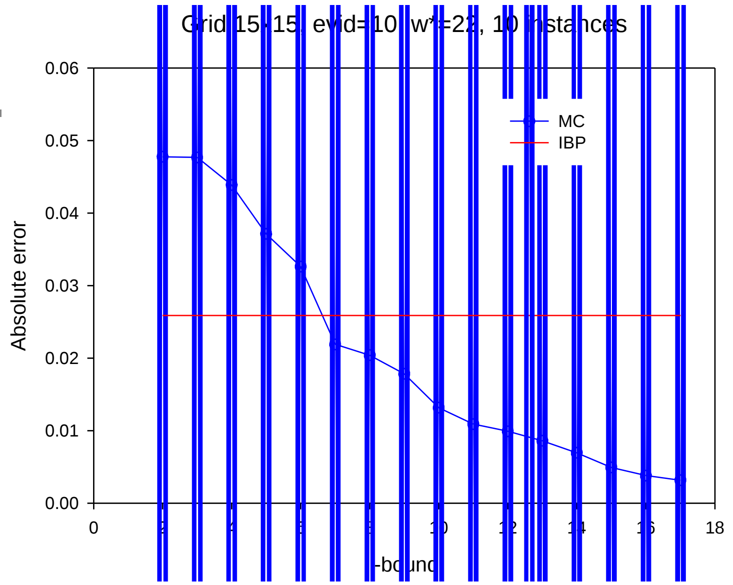


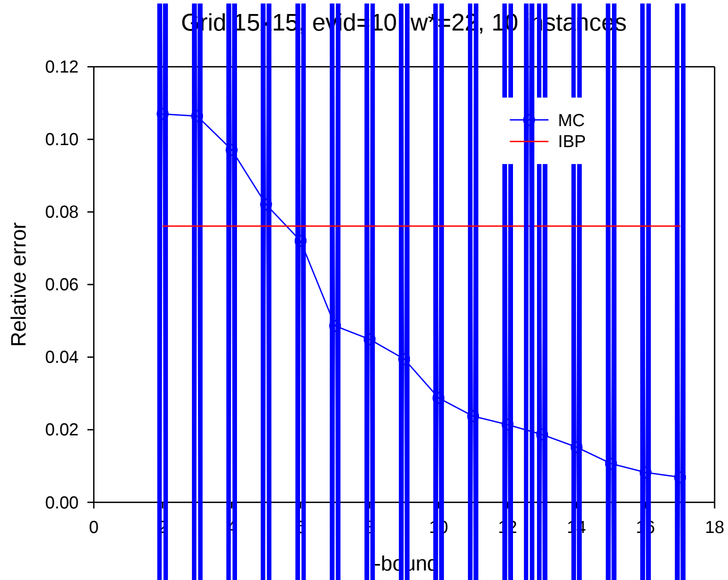
evidence=10

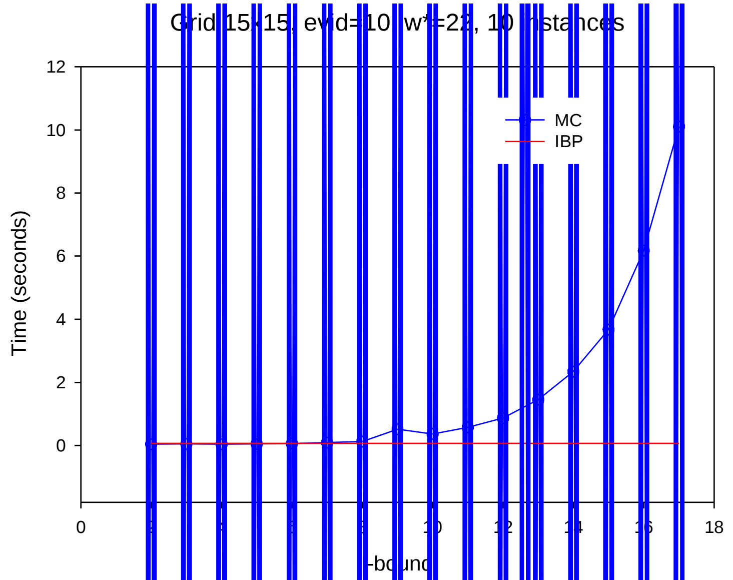
evidence=20

Grid 15x15 - 10 evidence

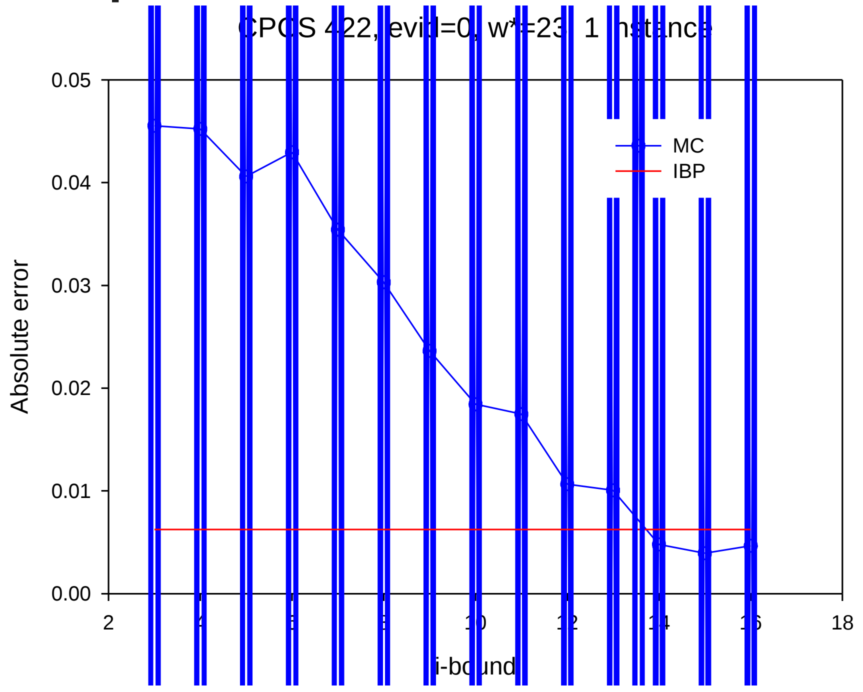


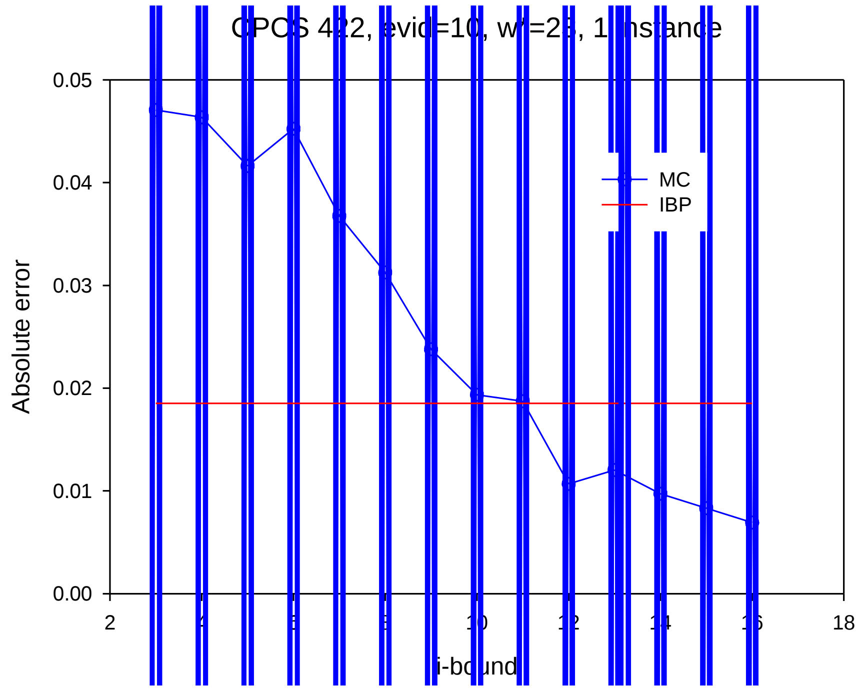






CPCS422 - Absolute error

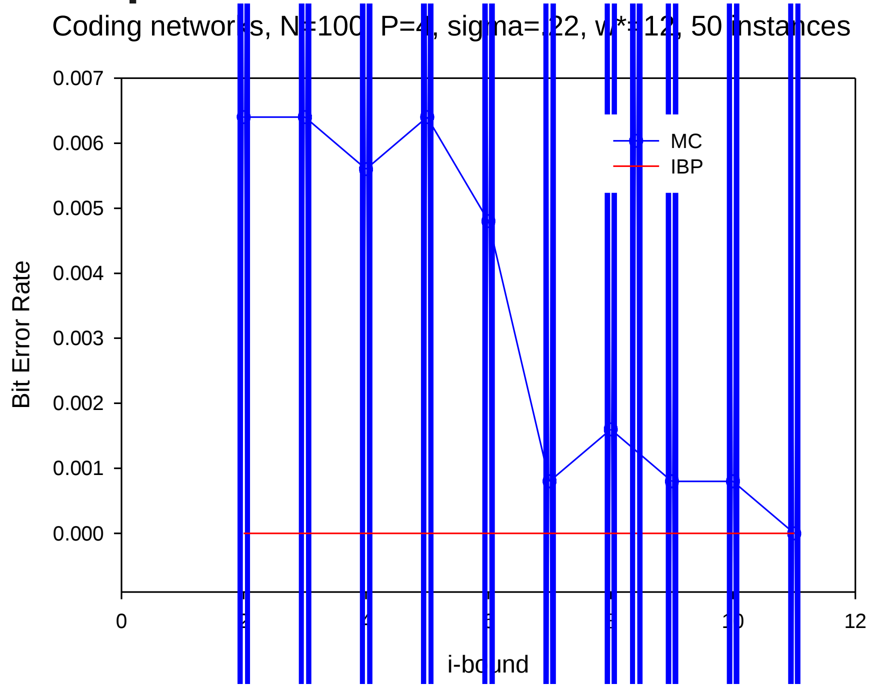


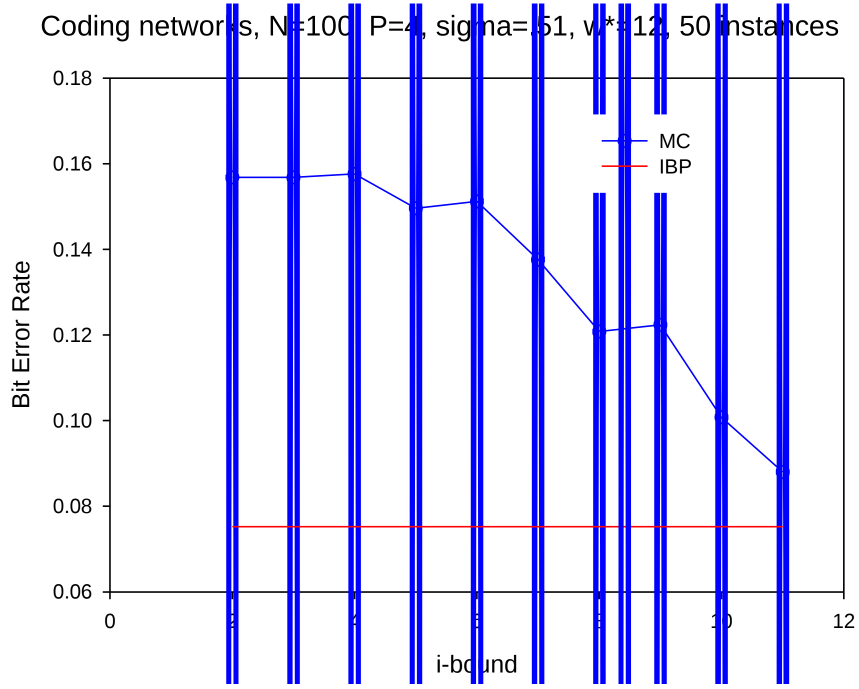


evidence=0

evidence=10

Coding networks - Bit Error Rate





sigma=0.22

sigma=.51

Mini-Clustering summary

- MC extends the partition based approximation from mini-buckets to general tree decompositions for the problem of belief updating
- Empirical evaluation demonstrates its effectiveness and superiority (for certain types of problems, with respect to the measures considered) relative to other existing algorithms

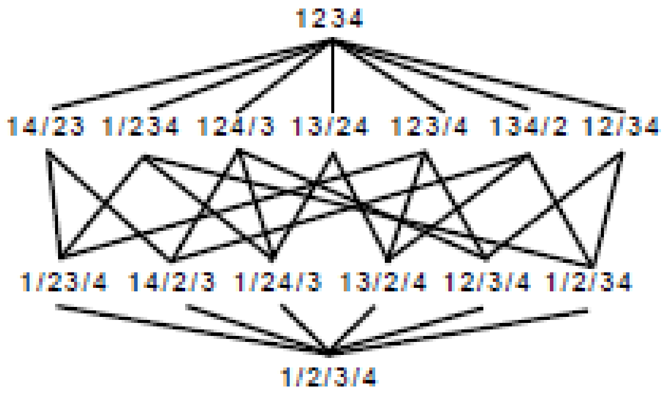
Heuristic for partitioning

Scope-based Partitioning Heuristic. The *scope-based* partition heuristic (SCP) aims at minimizing the number of mini-buckets in the partition by including in each minibucket as many functions as possible as long as the *i* bound is satisfied. First, single
function mini-buckets are decreasingly ordered according to their arity. Then, each minibucket is absorbed into the left-most mini-bucket with whom it can be merged.

The time and space complexity of Partition(*B*, *i*) , where *B* is the partitioned bucket, using the SCP heuristic is *O*(|*B*| log (|*B*|) + |*B*|^2) and *O*(*exp*(*i*)), respectively.

The scope-based
heuristic is is quite fast, its shortcoming is that it does not consider the actual information in the functions.

Content-based heuristics
(Rollon and Dechter 2010)



Partitioning lattice of bucket $\{f_1, f_2, f_3, f_4\}$.

- *Log relative error:*

$$RE(f,h) = \sum_t (\log(f(t)) - \log(h(t)))$$

- *Max log relative error:*

$$MRE(f,h) = \max_t \{\log(f(t)) - \log(h(t))\}$$

Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket

Iterative Join Graph Propagation

- Loopy Belief Propagation
  - Cyclic graphs
  - Iterative
  - Converges fast in practice (no guarantees though)
  - Very good approximations (e.g., turbo decoding, LDPC codes, SAT – survey propagation)

- Mini-Clustering(i)
  - Tree decompositions
  - Only two sets of messages (inward, outward)
  - Anytime behavior – can improve with more time by increasing the i-bound

- We want to combine:
  - Iterative virtues of Loopy BP
  - Anytime behavior of Mini-Clustering(i)

IJGP - The basic idea

- Apply Cluster Tree Elimination to any *join-graph*
- We commit to graphs that are *I-maps*
- Avoid cycles as long as I-mapness is not violated
- Result: use *minimal arc-labeled* join-graphs

Minimal arc-labeled join-graph

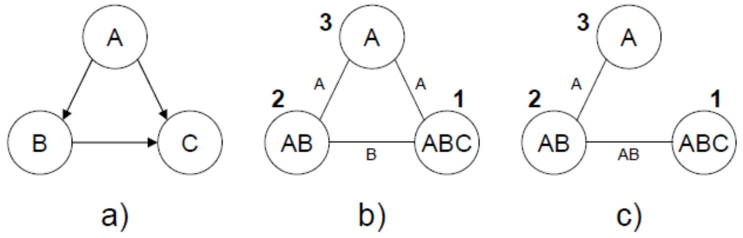


Figure 1.17: a) A belief network; b) A dual join-graph with singleton labels; c) A dual join-graph which is a join-tree

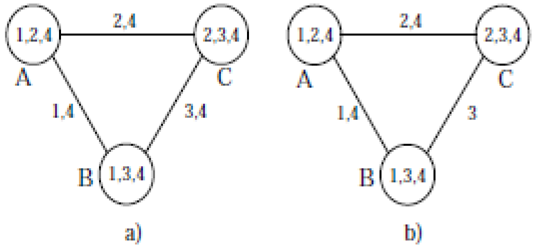
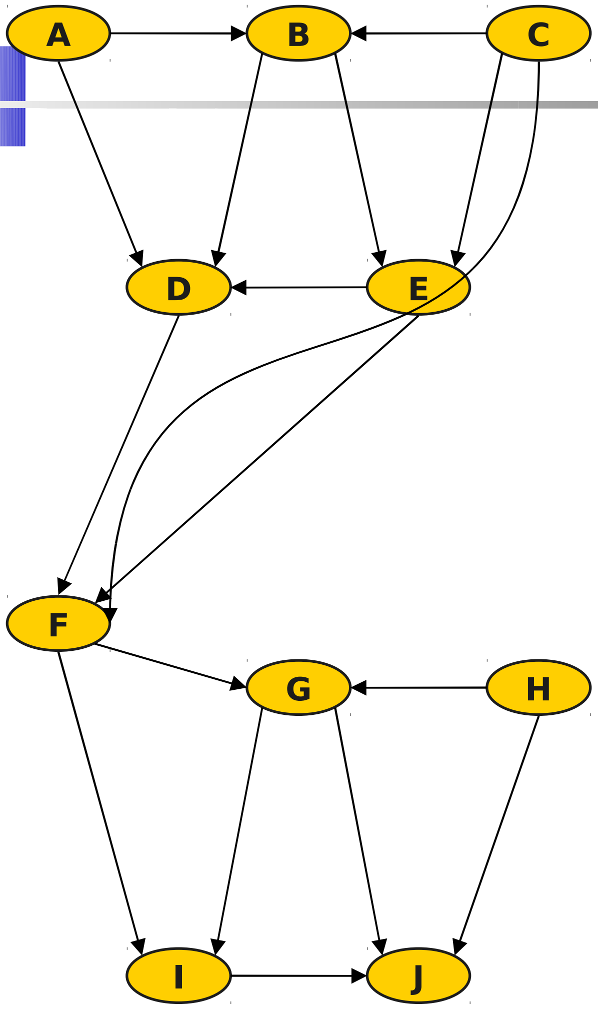
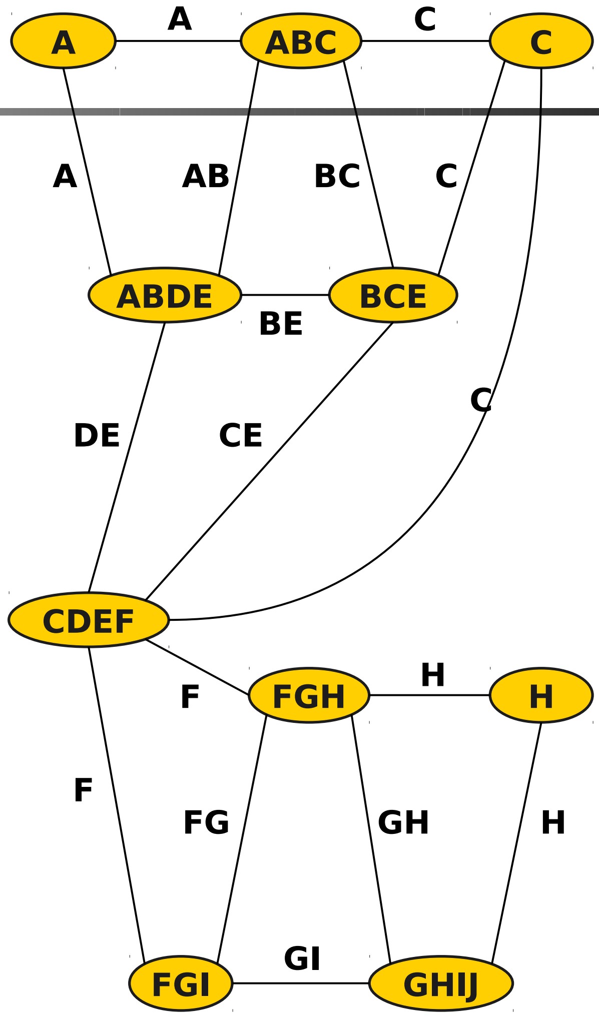


Figure 1.15: An arc-labeled decomposition

IJGP - Example

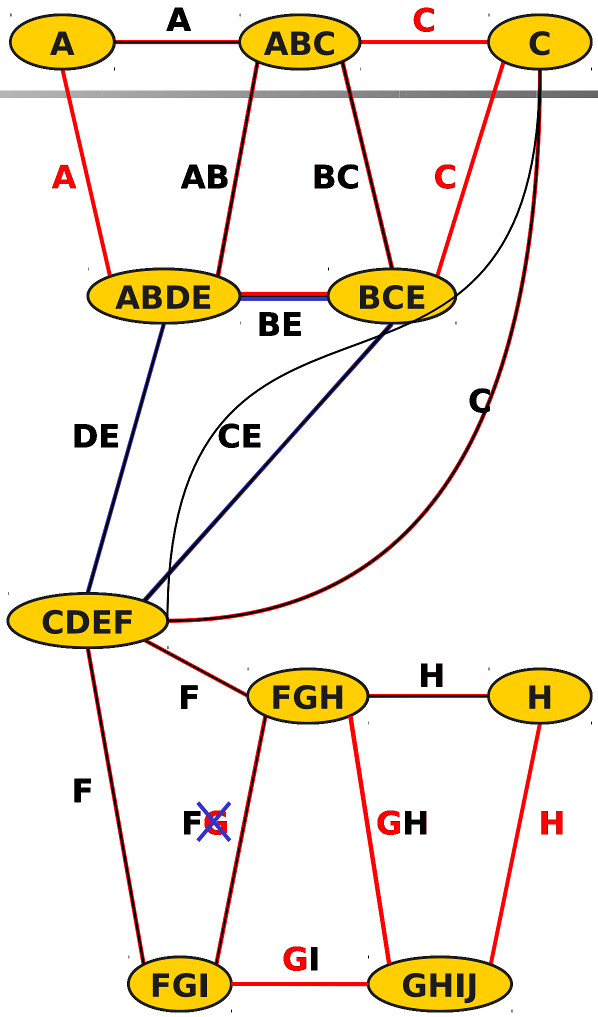


Belief network



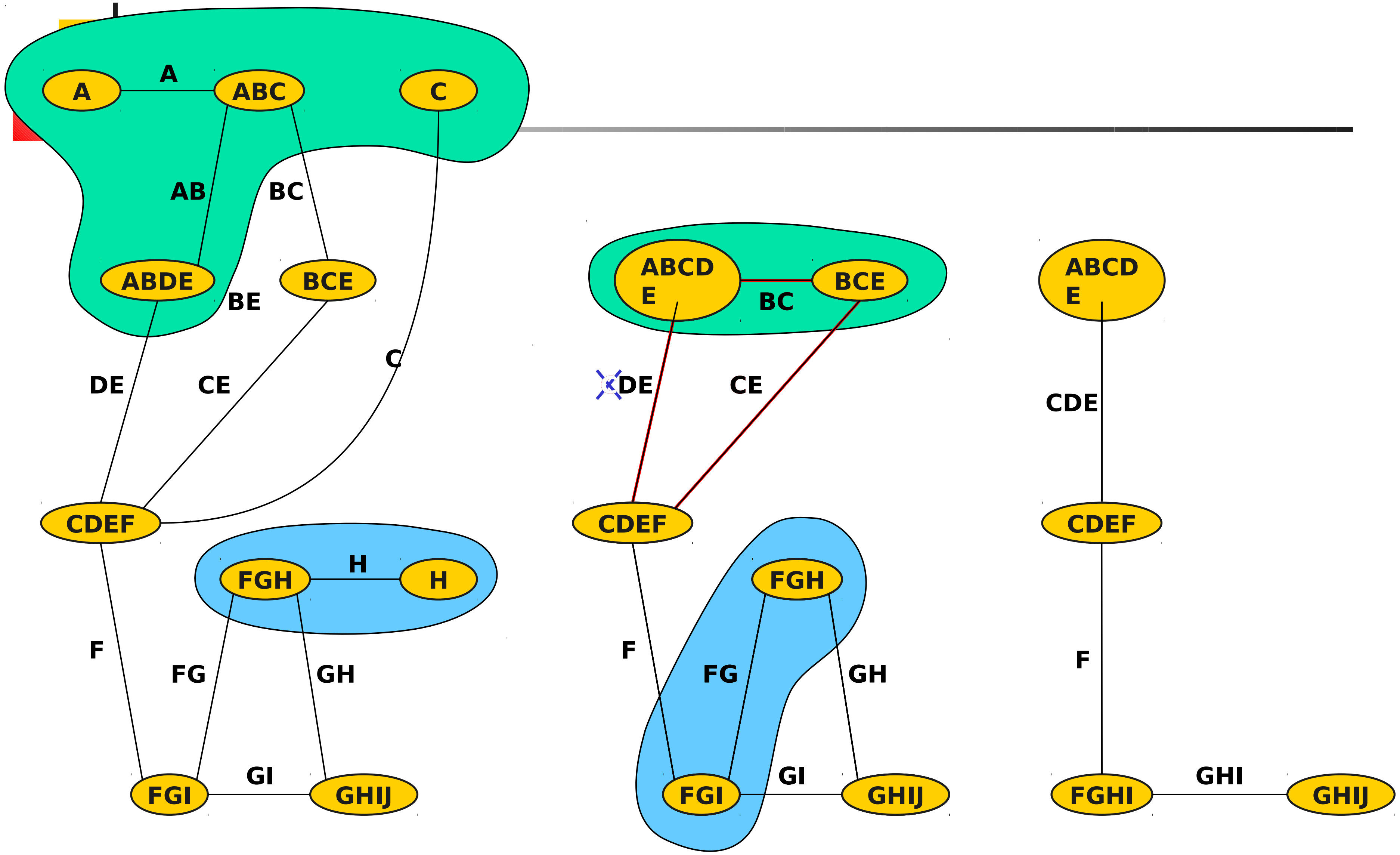
Loopy BP graph

Arc-Minimal Join-Graph

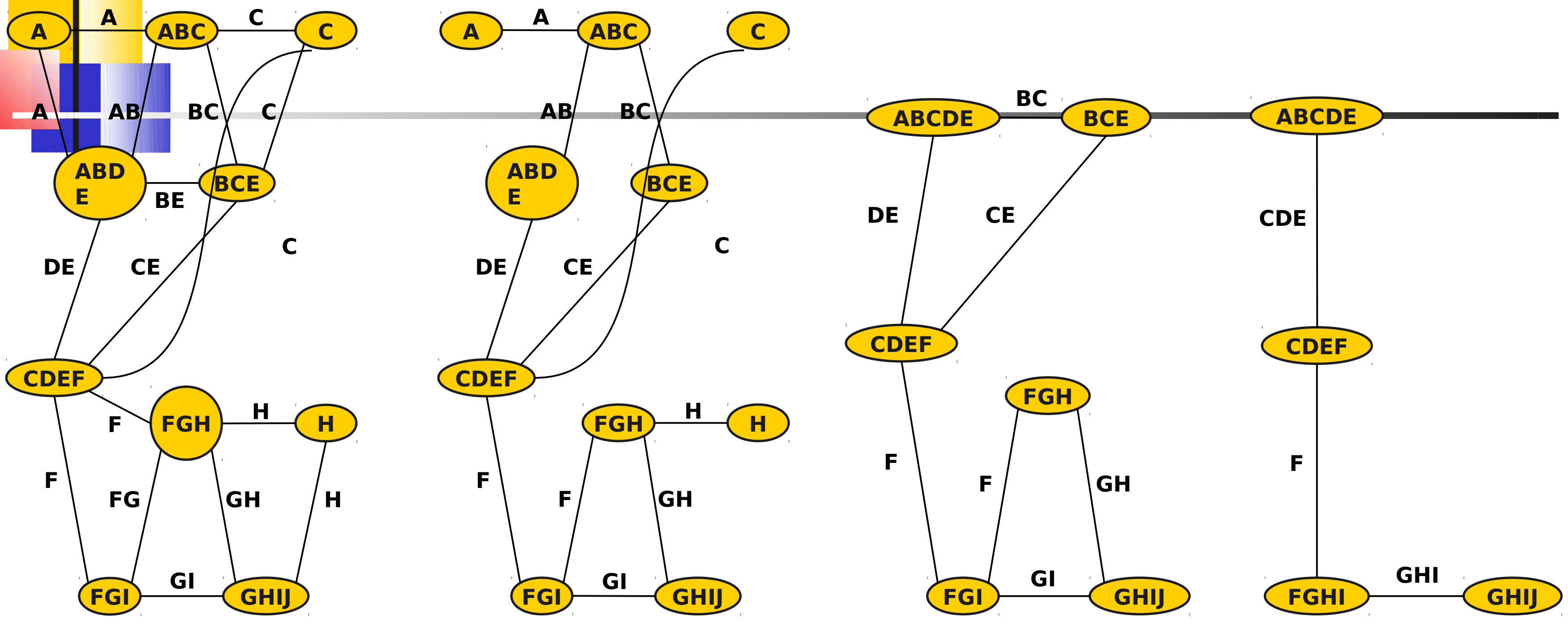


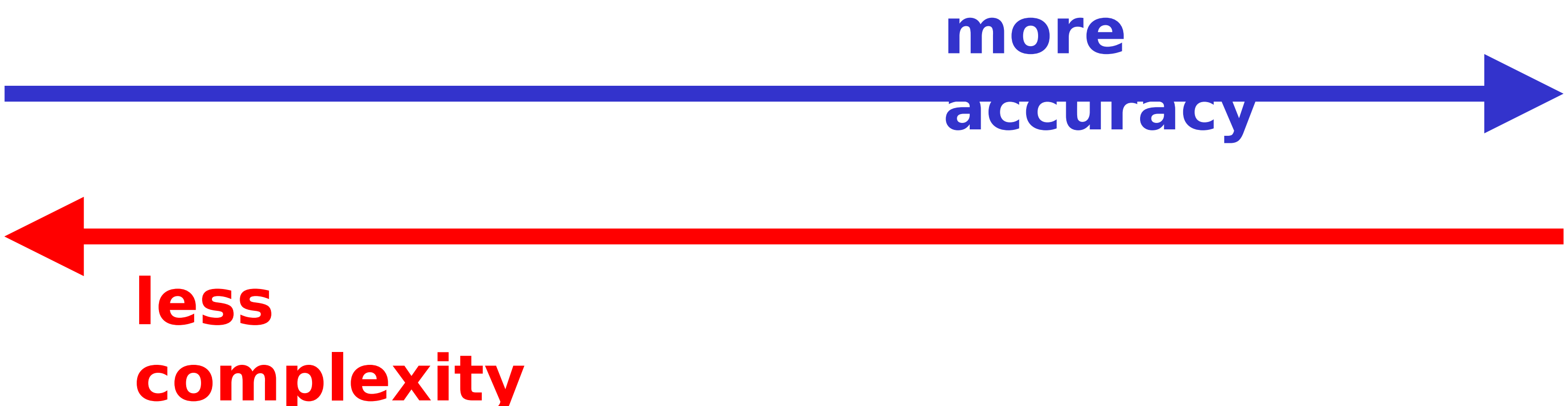
Arcs labeled with any single variable should form a TREE

Collapsing Clusters

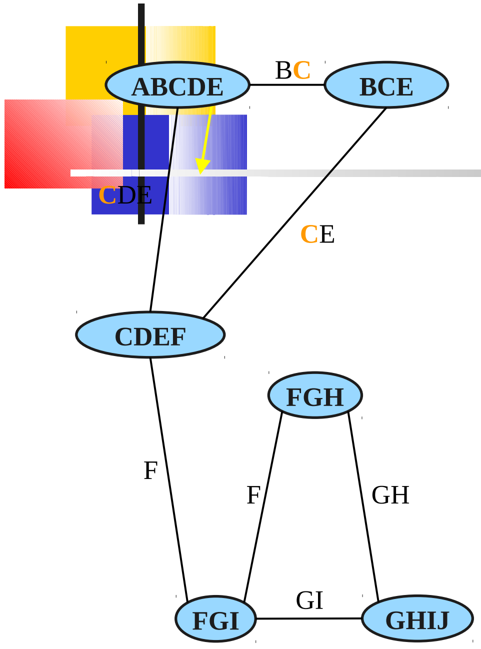


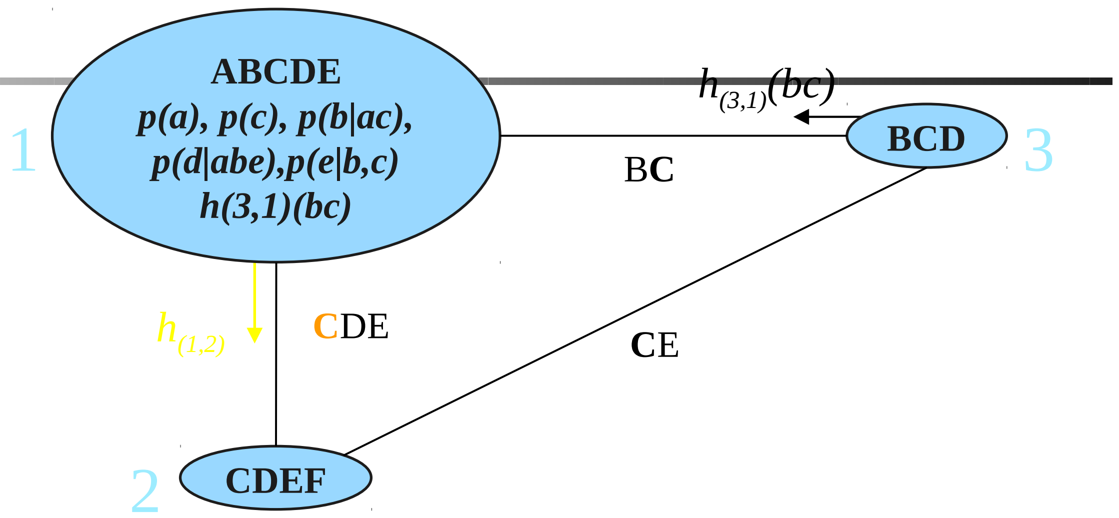
Join-Graphs


A
ABC
C
A
AB
BC
C
ABDE
BCE
BE
C
DE
CE
CDEF
FGH
H
F
F
FG
GH
H
FGI
GI
GHIJ
A
ABC
C
A
AB
BC
ABDE
BCE
C
DE
CE
CDEF
FGH
H
F
F
GH
FGI
GI
GHIJ
ABCDE
BC
BCE
DE
CE
CDEF
FGH
F
F
GH
FGI
GI
GHIJ
ABCDE
CDE
CDEF
F
FGHI
GHI
GHIJ


more accuracy
less complexity

Message propagation





Minimal arc-labeled:
sep(1,2)={D,E}
elim(1,2)={A,B,C}

$$h_{(1,2)}(de) = \sum_{a,b,c} p(a)p(c)p(b \mid ac)p(d \mid abe)p(e \mid bc)h_{(3,1)}(bc)$$

Non-minimal arc-labeled:
sep(1,2)={C,D,E}
elim(1,2)={A,B}

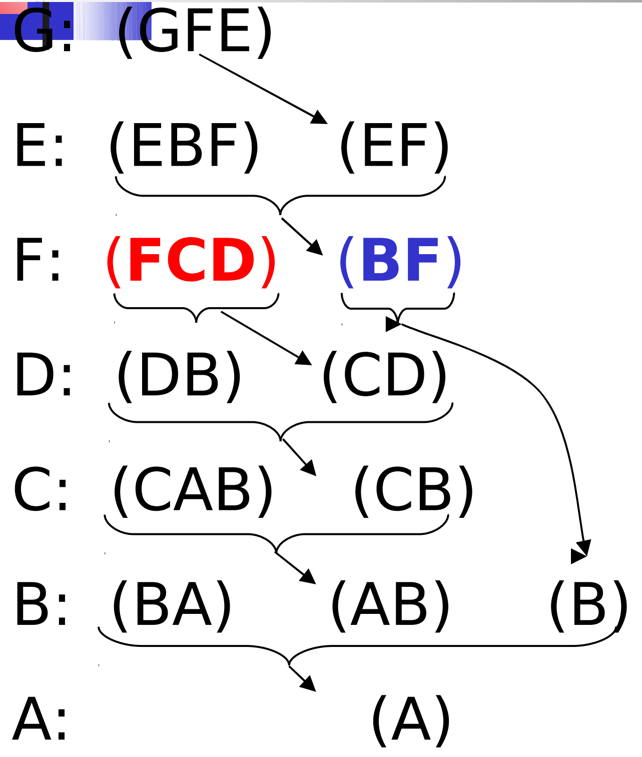
$$h_{(1,2)}(cde) = \sum_{a,b} p(a)p(c)p(b \mid ac)p(d \mid abe)p(e \mid bc)h_{(3,1)}(bc)$$

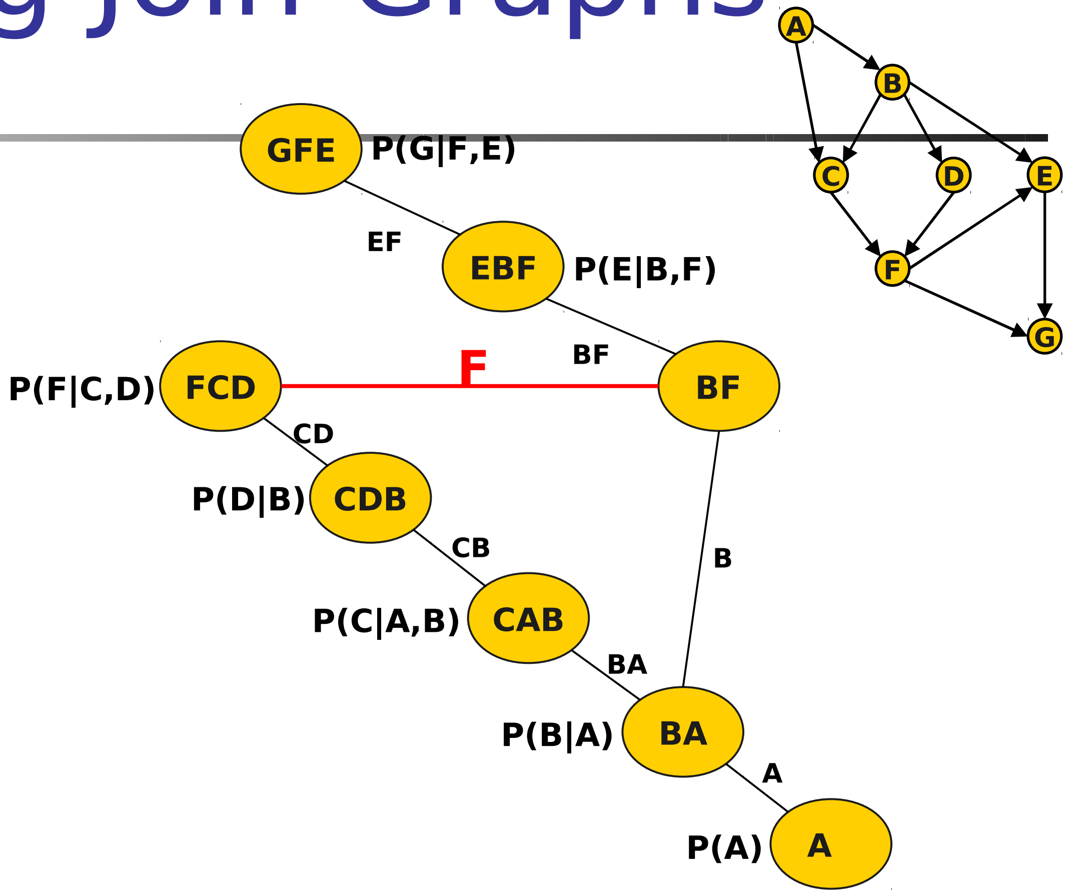
Bounded decompositions

- We want arc-labeled decompositions such that:
  - the cluster size (internal width) is bounded by *i* (the accuracy parameter)
  - the width of the decomposition as a graph (external width) is as small as possible

- Possible approaches to build decompositions:
  - partition-based algorithms - inspired by the mini-bucket decomposition
  - grouping-based algorithms

Constructing Join-Graphs





a) schematic mini-bucket(i), i=3 decomposition

b) arc-labeled join-graph

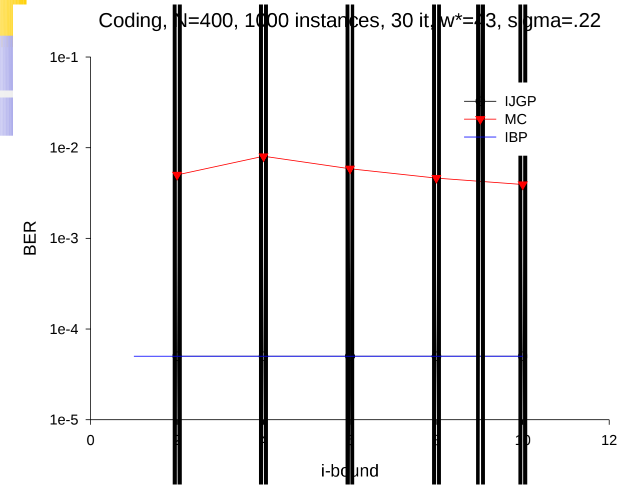
IJGP properties

- IJGP(*i*) applies BP to min arc-labeled join-graph, whose cluster size is bounded by *i*
- On join-trees IJGP finds exact beliefs
- IJGP is a Generalized Belief Propagation algorithm (Yedidia, Freeman, Weiss 2001)
- Complexity of one iteration:
  - time: $O(deg \bullet (n+N) \bullet d^{i+1})$
  - space: $O(N \bullet d^{\theta})$

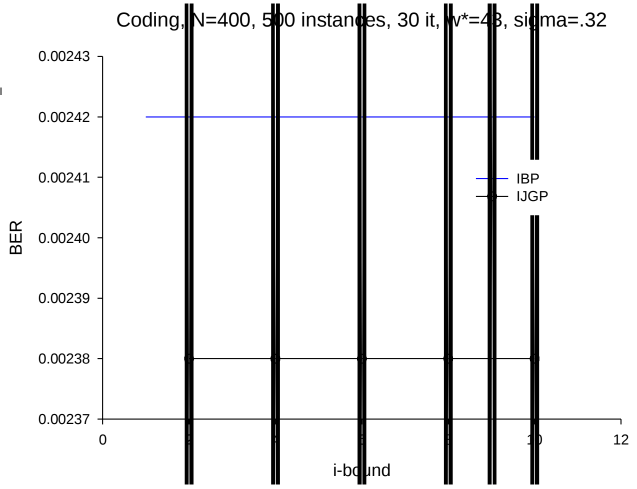
Empirical evaluation

- Measures:
  - Absolute error
  - Relative error
  - Kulbach-Leibler (KL) distance
  - Bit Error Rate
  - Time
- Algorithms:
  - Exact
  - IBP
  - MC
  - IJGP
- Networks (all variables are binary):
  - Random networks
  - Grid networks (MxM)
  - CPCS 54, 360, 422
  - Coding networks

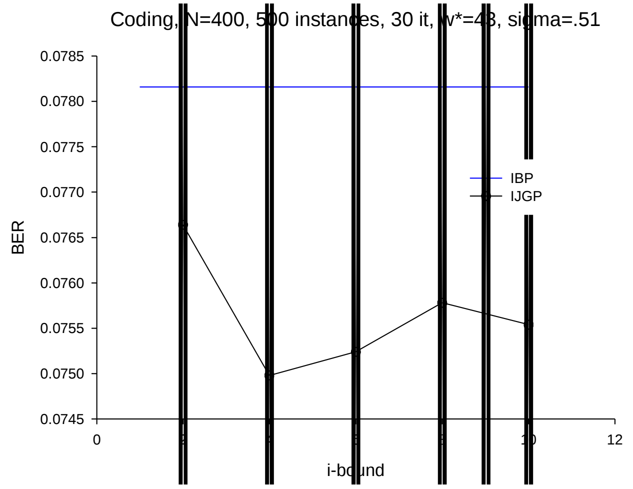
Coding networks - BER



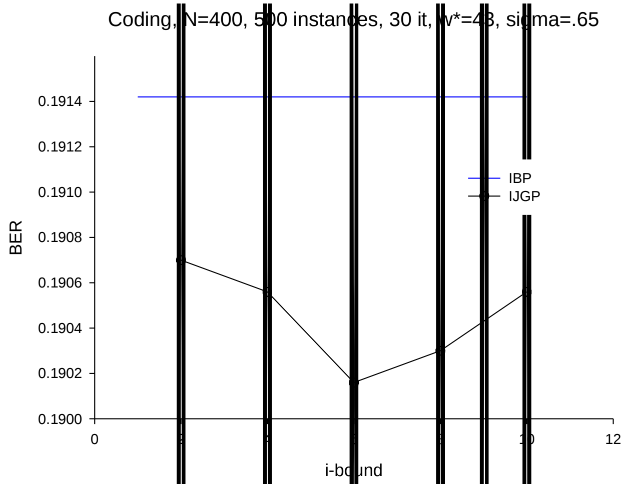
sigma=.22



sigma=.32

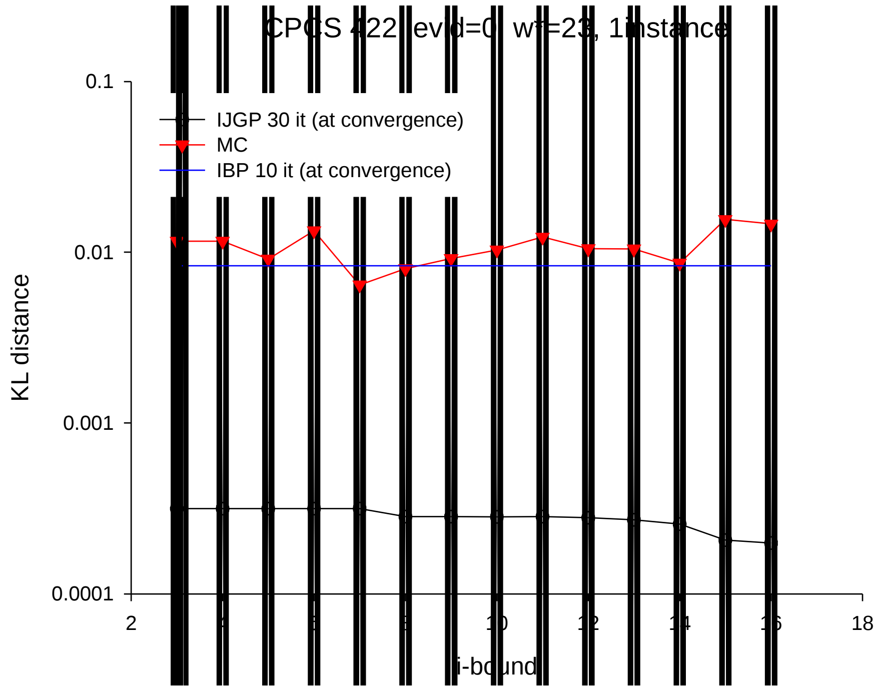


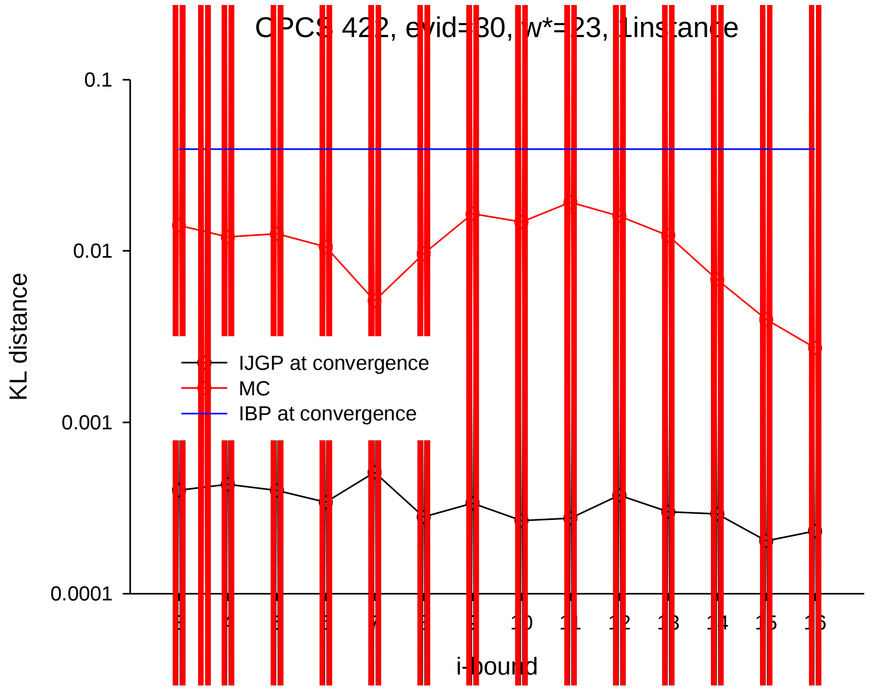
sigma=.51



sigma=.65

CPCS 422 – KL Distance

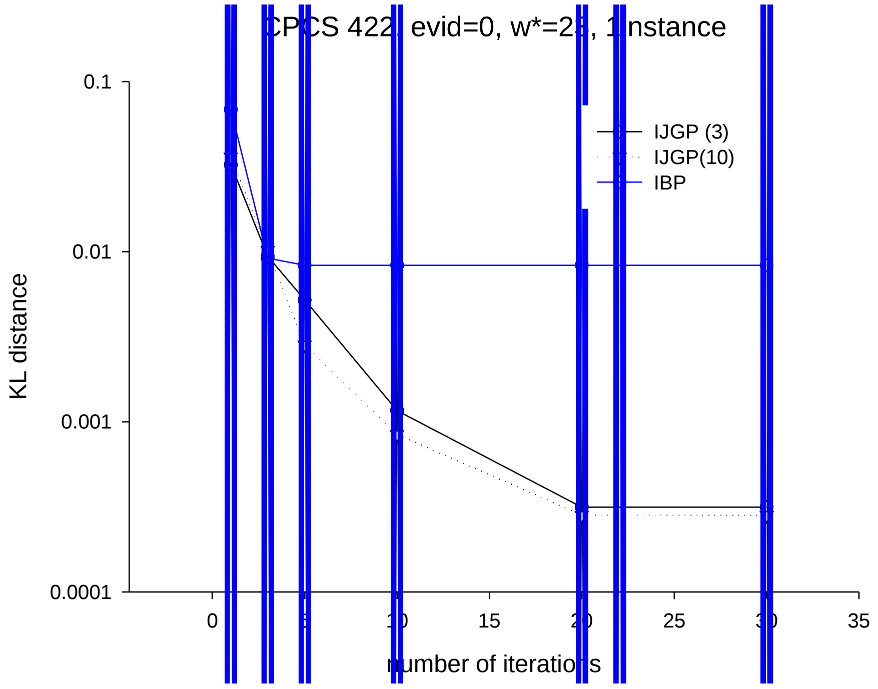




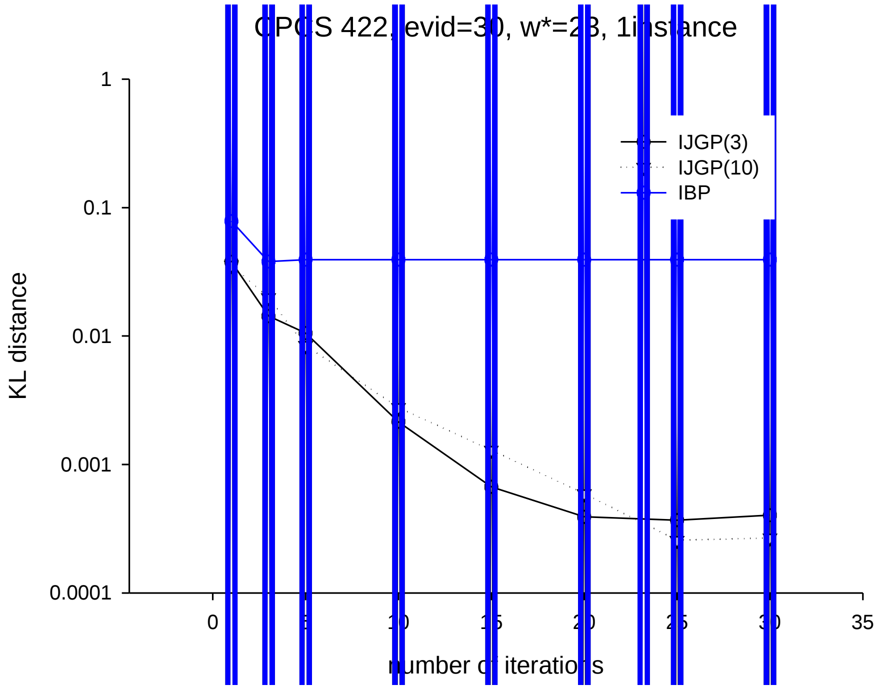
evidence=0

evidence=30

CPCS 422 – KL vs. Iterations

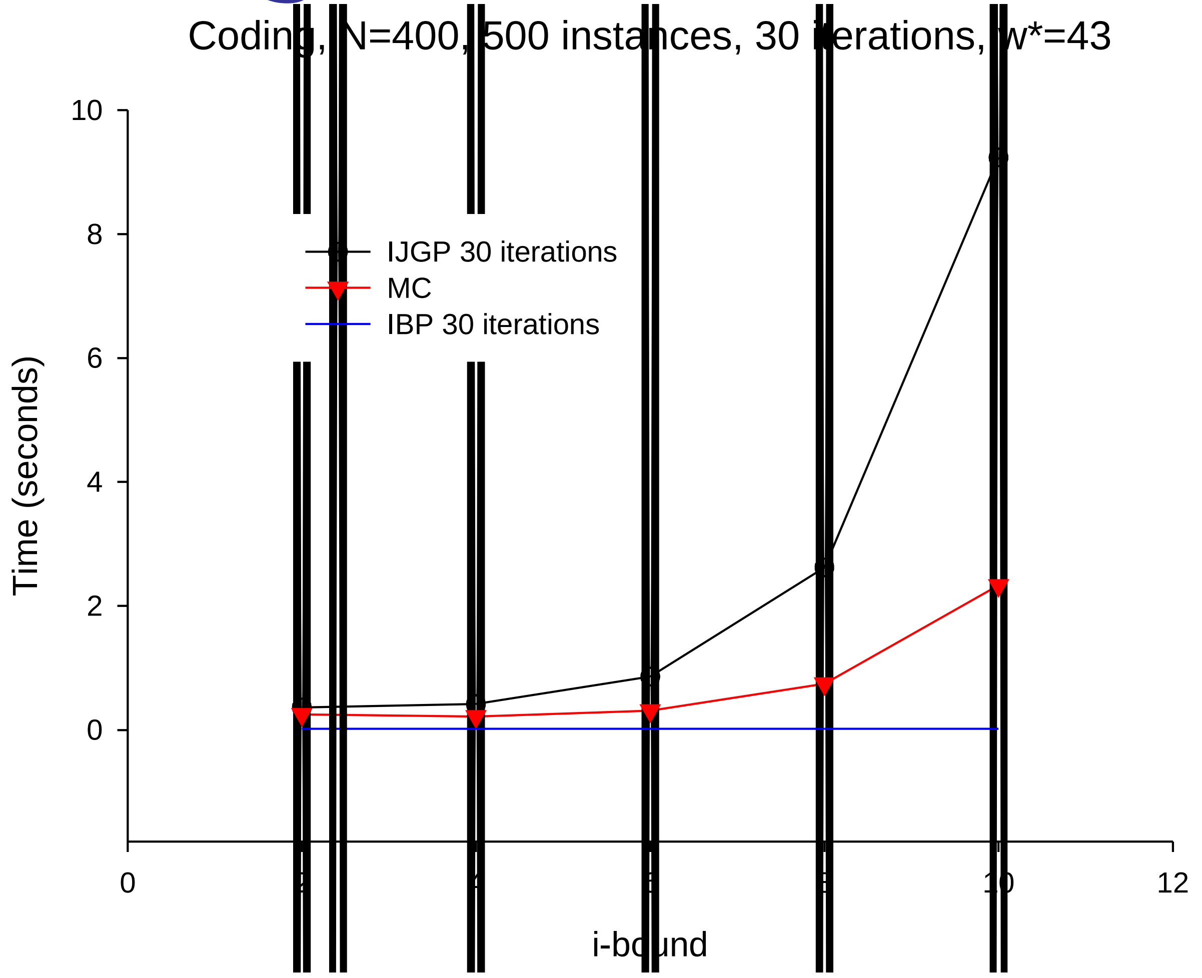


evidence=0



evidence=30

Coding networks - Time



More On the Power of Belief Propagation

- BP as local minima of KL distance
- BP's power from constraint propagation perspective.

More On the Power of Belief Propagation

- BP as local minima of KL distance
- BP's power from constraint propagation perspective.

The Kullback-Leibler Divergence

The Kullback-Leibler divergence (KL–divergence)

$$\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e})) = \sum_{\mathbf{x}} \Pr'(\mathbf{x}|\mathbf{e}) \log \frac{\Pr'(\mathbf{x}|\mathbf{e})}{\Pr(\mathbf{x}|\mathbf{e})}$$

- $\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e}))$ is non-negative
- equal to zero if and only if $\Pr'(\mathbf{X}|\mathbf{e})$ and $\Pr(\mathbf{X}|\mathbf{e})$ are equivalent.

The Kullback-Leibler Divergence

KL–divergence is not a true distance measure in that it is not symmetric. In general:

$$\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e})) \neq \mathrm{KL}(\Pr(\mathbf{X}|\mathbf{e}), \Pr'(\mathbf{X}|\mathbf{e})).$$

- $\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e}))$ weighting the KL–divergence by the approximate distribution $\Pr'$
- We shall indeed focus on the KL–divergence weighted by the approximate distribution as it has some useful computational properties.

The Kullback-Leibler Divergence

Let $\Pr(\mathbf{X})$ be a distribution induced by a Bayesian network $\mathcal{N}$ having families $X\mathbf{U}$

The KL–divergence between $\Pr$ and another distribution $\Pr'$ can be written as a sum of three components:

$$\begin{aligned}&\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e})) \\ &\quad = \quad -\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) - \sum_{X\mathbf{U}} \mathrm{AVG}'(\log \lambda_{\mathbf{e}}(X)\Theta_{X|\mathbf{U}}) + \log \Pr(\mathbf{e}),\end{aligned}$$

where

- $\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) = -\sum_{\mathbf{x}} \Pr'(\mathbf{x}|\mathbf{e}) \log \Pr'(\mathbf{x}|\mathbf{e})$ is the entropy of the conditioned approximate distribution $\Pr'(\mathbf{X}|\mathbf{e})$.
- $\mathrm{AVG}'(\log \lambda_{\mathbf{e}}(X)\Theta_{X|\mathbf{U}}) = \sum_{x\mathbf{u}} \Pr'(x\mathbf{u}|\mathbf{e}) \log \lambda_{\mathbf{e}}(x)\theta_{x|\mathbf{u}}$ is a set of expectations over the original network parameters weighted by the conditioned approximate distribution.

The Kullback-Leibler Divergence

A distribution $\Pr'(\mathbf{X}|\mathbf{e})$ minimizes the KL-divergence $\mathrm{KL}(\Pr'(\mathbf{X}|\mathbf{e}), \Pr(\mathbf{X}|\mathbf{e}))$ if it maximizes

$$\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) + \sum_{X\mathbf{U}} \mathrm{AVG}'(\log \lambda_{\mathbf{e}}(X)\Theta_{X|\mathbf{U}})$$

Competing properties of $\Pr'(\mathbf{X}|\mathbf{e})$ that minimize the KL–divergence:

- $\Pr'(\mathbf{X}|\mathbf{e})$ should match the original distribution by giving more weight to more likely parameters $\lambda_{\mathbf{e}}(x)\theta_{x|\mathbf{u}}$ (i.e, maximize the expectations).
- $\Pr'(\mathbf{X}|\mathbf{e})$ should not favor unnecessarily one network instantiation over another by being evenly distributed (i.e., maximize the entropy).

Optimizing the KL-Divergence

The approximations computed by IBP are based on assuming an approximate distribution $\Pr'(\mathbf{X})$ that factors as follows:

$$\Pr'(\mathbf{X}|\mathbf{e}) = \prod_{X\mathbf{U}} \frac{\Pr'(X\mathbf{U}|\mathbf{e})}{\prod_{U\in\mathbf{U}} \Pr'(U|\mathbf{e})}$$

- This choice of $\Pr'(\mathbf{X}|\mathbf{e})$ is expressive enough to describe distributions $\Pr(\mathbf{X}|\mathbf{e})$ induced by polytree networks $\mathcal{N}$
- In the case where $\mathcal{N}$ is not a polytree, then we are simply trying to fit $\Pr(\mathbf{X}|\mathbf{e})$ into an approximation $\Pr'(\mathbf{X}|\mathbf{e})$ as if it were generated by a polytree network.
- The entropy of distribution $\Pr'(\mathbf{X}|\mathbf{e})$ can be expressed as:

$$\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) = -\sum_{X\mathbf{U}} \sum_{x\mathbf{u}} \Pr'(x\mathbf{u}|\mathbf{e}) \log \frac{\Pr'(x\mathbf{u}|\mathbf{e})}{\prod_{u\sim\mathbf{u}} \Pr'(u|\mathbf{e})}$$

Optimizing the KL-Divergence

Let $\Pr(\mathbf{X})$ be a distribution induced by a Bayesian network $\mathcal{N}$ having families $X\mathbf{U}$. Then IBP messages are a fixed point if and only if IBP marginals $\mu_u = BEL(u)$ and $\mu_{x\mathbf{u}} = BEL(x\mathbf{u})$ are a stationary point of:

$$\begin{aligned}&\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) + \sum_{X\mathbf{U}} \mathrm{AVG}'(\log \lambda_{\mathbf{e}}(X)\Theta_{X|\mathbf{U}}) \\ = \quad & -\sum_{X\mathbf{U}}\sum_{x\mathbf{u}} \mu_{x\mathbf{u}} \log \frac{\mu_{x\mathbf{u}}}{\prod_{u\sim\mathbf{u}} \mu_u} + \sum_{X\mathbf{U}}\sum_{x\mathbf{u}} \mu_{x\mathbf{u}} \log \lambda_{\mathbf{e}}(x)\theta_{x|\mathbf{u}},\end{aligned}$$

under normalization constraints:

$$\sum_u \mu_u = \sum_{x\mathbf{u}} \mu_{x\mathbf{u}} = 1$$

for each family $X\mathbf{U}$ and parent U, and under consistency constraints:

$$\sum_{x\mathbf{u}\sim y} \mu_{x\mathbf{u}} = \mu_y$$

for each family instantiation $x\mathbf{u}$ and value y of family member $Y \in X\mathbf{U}$.

Optimizing the KL-Divergence

- IBP fixed points are stationary points of the KL–divergence: they may only be local minima, or they may not be minima.
- When IBP performs well, it will often have fixed points that are indeed minima of the KL–divergence.
- For problems where IBP does not behave as well, we will next seek approximations Pr' whose factorizations are more expressive than that of the polytree-based factorization.

Generalized Belief Propagation

If a distribution $\Pr'$ has the form:

$$\Pr'(\mathbf{X}|\mathbf{e}) = \frac{\prod_{\mathbf{C}} \Pr'(\mathbf{C}|\mathbf{e})}{\prod_{\mathbf{S}} \Pr'(\mathbf{S}|\mathbf{e})},$$

then its entropy has the form:

$$\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) = \sum_{\mathbf{C}} \mathrm{ENT}'(\mathbf{C}|\mathbf{e}) - \sum_{\mathbf{S}} \mathrm{ENT}'(\mathbf{S}|\mathbf{e}).$$

When the marginals $\Pr'(\mathbf{C}|\mathbf{e})$ and $\Pr'(\mathbf{S}|\mathbf{e})$ are readily available, the ENT component of the KL–divergence can be computed efficiently.

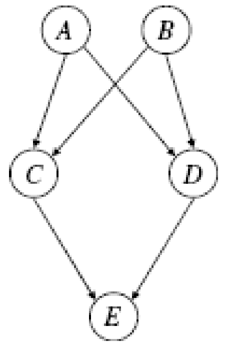
Joingraphs

While a jointree induces an exact factorization of a distribution, a joingraph G induces an approximate factorization:

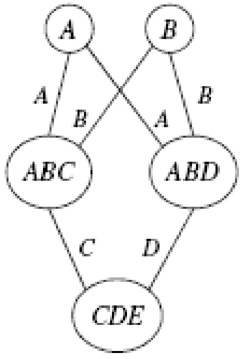
$$\Pr'(\mathbf{X}|\mathbf{e}) = \frac{\prod_i \Pr'(\mathbf{C}_i|\mathbf{e})}{\prod_{ij} \Pr'(\mathbf{S}_{ij}|\mathbf{e})}$$

which is a product of cluster marginals over a product of separator marginals. When the joingraph corresponds to a jointree, the above factorization will be exact.

Joingraphs



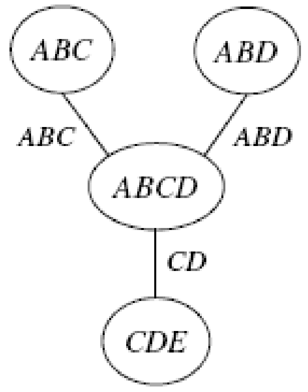
Bayesian network



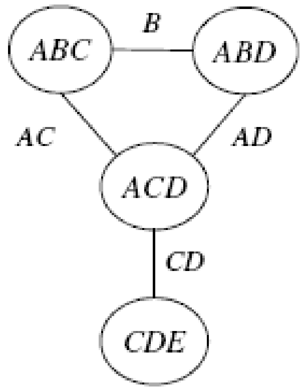
dual joingraph

A dual joingraph leads to the factorization used by IBP.

Joingraphs



jointree

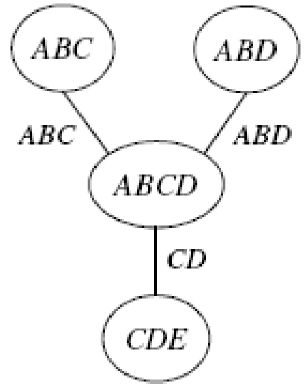


joingraph

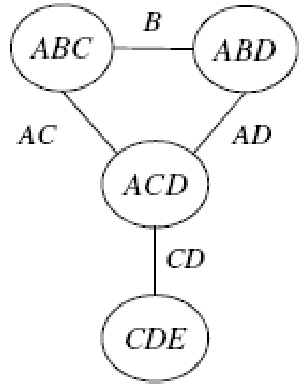
The jointree induces the following factorization, which is exact:

$$\Pr'(\mathbf{X}|\mathbf{e}) = \frac{\Pr'(ABC|\mathbf{e})\Pr'(ABD|\mathbf{e})\Pr'(ABCD|\mathbf{e})\Pr'(CDE|\mathbf{e})}{\Pr'(ABC|\mathbf{e})\Pr'(ABD|\mathbf{e})\Pr'(CD|\mathbf{e})}$$

Joingraphs



jointree



joingraph

The joingraph induces the following factorization:

$$\mathrm{Pr}'(\mathbf{X}|\mathbf{e}) = \frac{\mathrm{Pr}'(ABC|\mathbf{e})\mathrm{Pr}'(ABD|\mathbf{e})\mathrm{Pr}'(ACD|\mathbf{e})\mathrm{Pr}'(CDE|\mathbf{e})}{\mathrm{Pr}'(B|\mathbf{e})\mathrm{Pr}'(AC|\mathbf{e})\mathrm{Pr}'(AD|\mathbf{e})\mathrm{Pr}'(CD|\mathbf{e})}$$

terative Joingraph Propagation

Computing cluster marginals $\mu_{\mathbf{c}_i} = \mathrm{Pr}'(\mathbf{c}_i|\mathbf{e})$ and separator marginals $\mu_{\mathbf{s}_{ij}} = \mathrm{Pr}'(\mathbf{s}_{ij}|\mathbf{e})$ that minimize the KL–divergence between $\mathrm{Pr}'(\mathbf{X}|\mathbf{e})$ and $\mathrm{Pr}(\mathbf{X}|\mathbf{e})$

This optimization problem can be solved using a generalization of IBP, called iterative joingraph propagation (IJGP), which is a message passing algorithm that operates on a joingraph.

Iterative Joingraph Propagation

```
IJGP($G$, $\Phi$)
input:
  $G$:  a joingraph
  $\Phi$:  factors assigned to clusters of $G$

output: approximate marginal $BEL(\mathbf{C}_i)$ for each node $i$ in the joingraph $G$.

main:
1: $t \leftarrow 0$
2: initialize all messages $M_{ij}^t$ (uniformly)
3: while messages have not converged do
4:     $t \leftarrow t + 1$
5:     for each joingraph edge $i$–$j$ do
6:         $M_{ij}^t \leftarrow \eta \sum_{\mathbf{C}_i \setminus \mathbf{S}_{ij}} \Phi_i \prod_{k \neq j} M_{ki}^{t-1}$
7:         $M_{ji}^t \leftarrow \eta \sum_{\mathbf{C}_j \setminus \mathbf{S}_{ij}} \Phi_j \prod_{k \neq i} M_{kj}^{t-1}$
8:     end for
9: end while
10: return $BEL(\mathbf{C}_i) \leftarrow \eta \, \Phi_i \prod_k M_{ki}^t$ for each node $i$
```

terative Joingraph Propagation

Let $\Pr(\mathbf{X})$ be a distribution induced by a Bayesian network $\mathcal{N}$ having families $X\mathbf{U}$, and let $\mathbf{C}_i$ and $\mathbf{S}_{ij}$ be the clusters and separators of a joingraph for $\mathcal{N}$.
Then messages M_{ij} are a fixed point of `IJGP` if and only if `IJGP` marginals $\mu_{\mathbf{c}_i} = BEL(\mathbf{c}_i)$ and $\mu_{\mathbf{s}_{ij}} = BEL(\mathbf{s}_{ij})$ are a stationary point of:

$$\begin{aligned} &\mathrm{ENT}'(\mathbf{X}|\mathbf{e}) + \sum_{\mathbf{C}_i} \mathrm{AVG}'(\log \Phi_i) \\ = \quad & -\sum_{\mathbf{C}_i}\sum_{\mathbf{c}_i} \mu_{\mathbf{c}_i} \log \mu_{\mathbf{c}_i} + \sum_{\mathbf{S}_{ij}}\sum_{\mathbf{s}_{ij}} \mu_{\mathbf{s}_{ij}} \log \mu_{\mathbf{s}_{ij}} + \sum_{\mathbf{C}_i}\sum_{\mathbf{c}_i} \mu_{\mathbf{c}_i} \log \Phi_i(\mathbf{c}_i), \end{aligned}$$

under normalization constraints:

$$\sum_{\mathbf{c}_i} \mu_{\mathbf{c}_i} = \sum_{\mathbf{s}_{ij}} \mu_{\mathbf{s}_{ij}} = 1$$

for each cluster $\mathbf{C}_i$ and separator $\mathbf{S}_{ij}$, and under consistency constraints:

$$\sum_{\mathbf{c}_i \sim \mathbf{s}_{ij}} \mu_{\mathbf{c}_i} = \mu_{\mathbf{s}_{ij}} = \sum_{\mathbf{c}_j \sim \mathbf{s}_{ij}} \mu_{\mathbf{c}_j}$$

for each separator $\mathbf{S}_{ij}$ and neighboring clusters $\mathbf{C}_i$ and $\mathbf{C}_j$.

Summary of IJGP so far

A spectrum of approximations.

IBP: results from applying IJGP to the dual joingraph.

Jointree algorithm: results from applying IJGP to a jointree (as a joingraph).

In between these two ends, we have a spectrum of joingraphs and corresponding factorizations, where IJGP seeks stationary points of the KL–divergence between these factorizations and the original distribution.

More On the Power of Belief Propagation

- BP as local minima of KL distance
- BP’s power from constraint propagation perspective.

Inference Power of Loopy BP

- Comparison with iterative algorithms in constraint networks
- Zero-beliefs assignments $\Leftrightarrow$ inconsistent
- ε-small beliefs – experimental study

Constraint networks

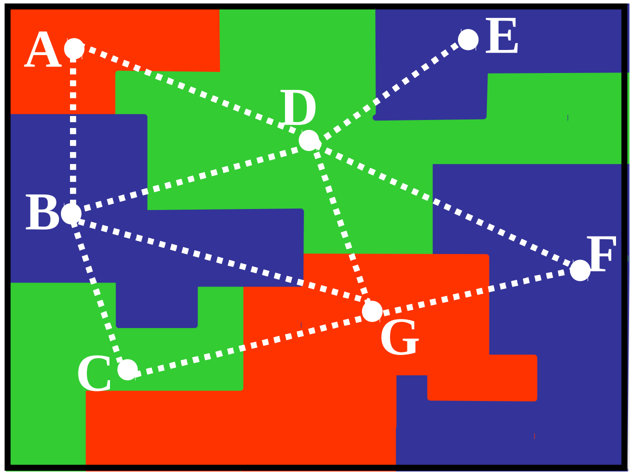
Map coloring

Variables: countries (A B C etc.)

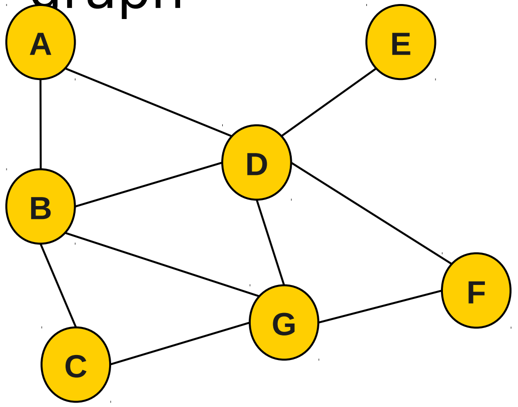
Values: colors (red green blue)

Constraints: $A \neq B$, $A \neq D$, $D \neq E$, *etc.*

A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red

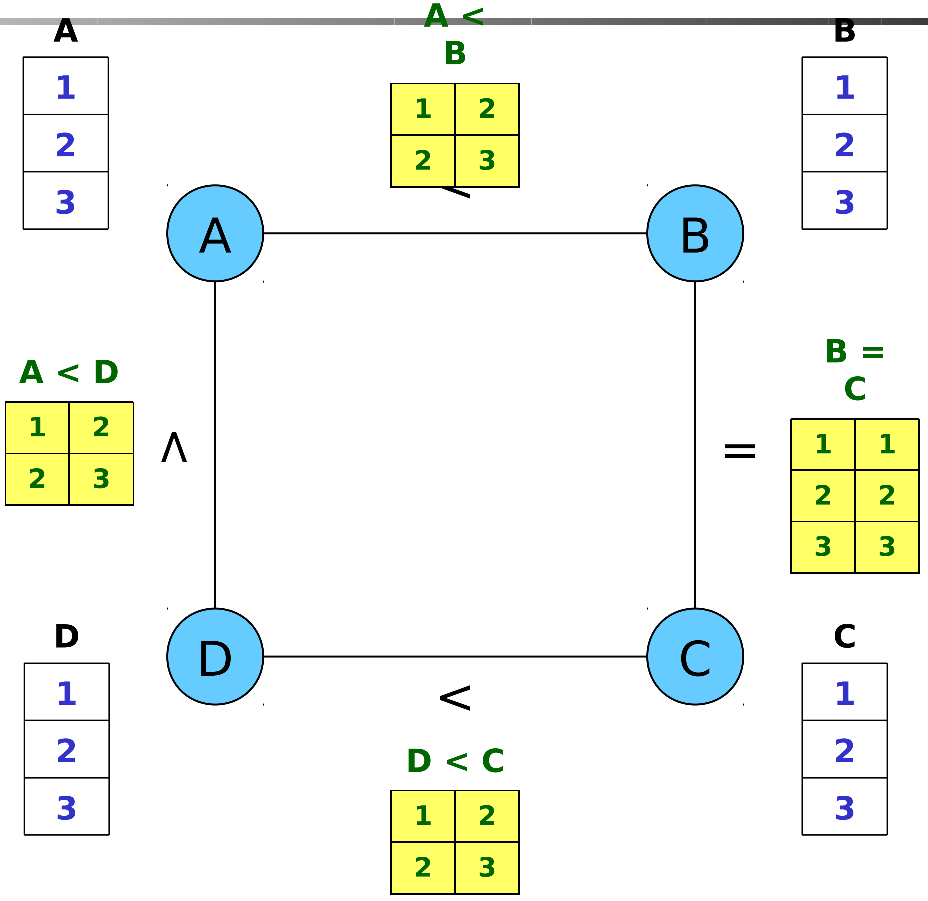


Constraint graph

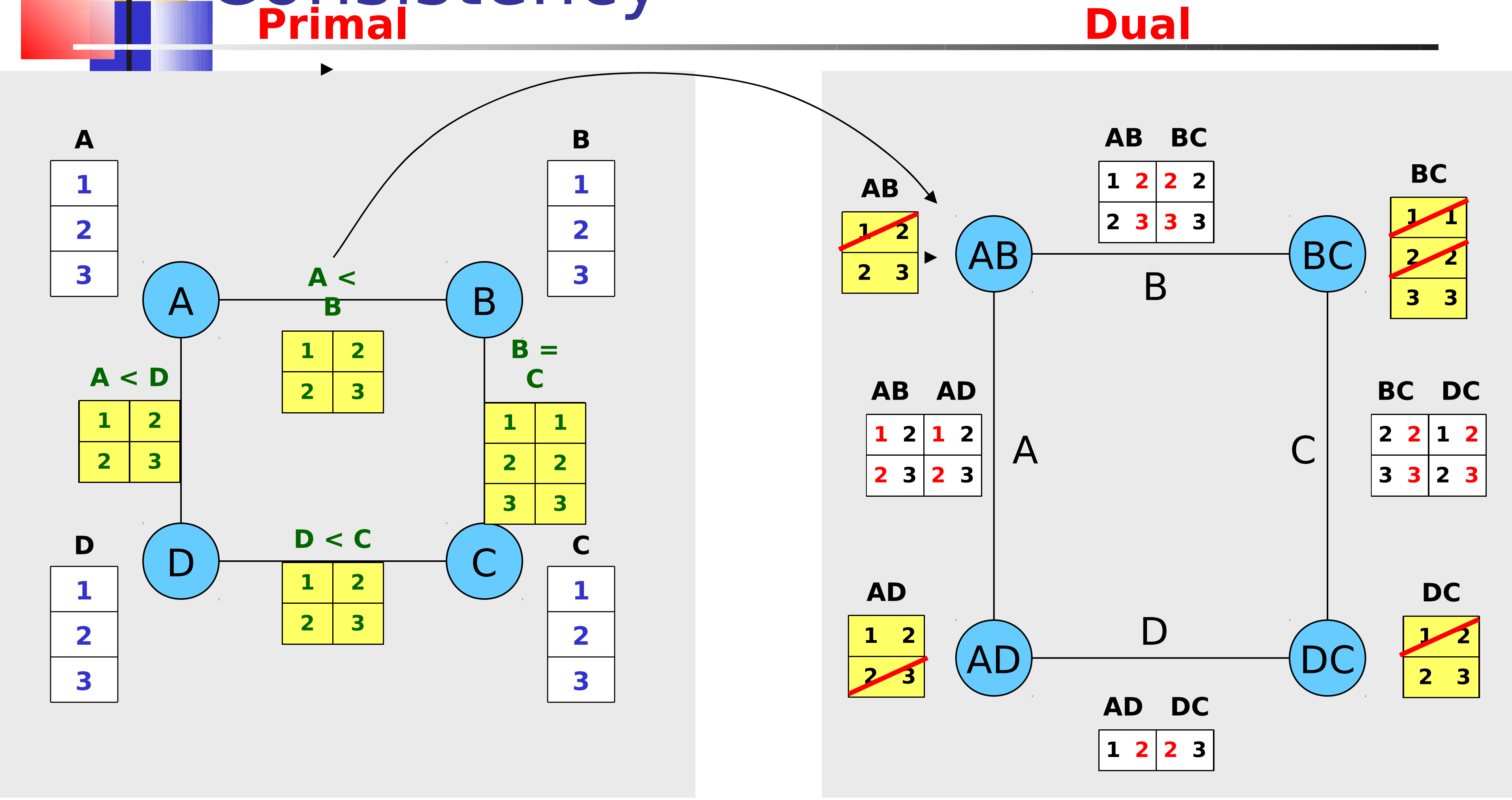


Arc-consistency

- Sound
- Incomplete
- Always converges (polynomial)



Relational Distributed Arc-Consistency

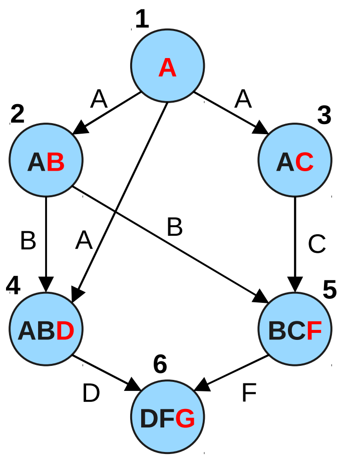


Flattening the Bayesian Network

A	P(A)
1	.2
2	.5
3	.3
…	0

A	B	P(B|A)
1	2	.3
1	3	.7
2	1	.4
2	3	.6
3	1	.1
3	2	.9
…	…	0

A	C	P(C|A)
1	2	1
3	2	1
…	…	0



A	B	D	P(D|A,B)
1	2	3	1
1	3	2	1
2	1	3	1
2	3	1	1
3	1	2	1
3	2	1	1
…	…	…	0

B	C	F	P(F|B,C)
1	2	3	1
3	2	1	1
…	…	…	0

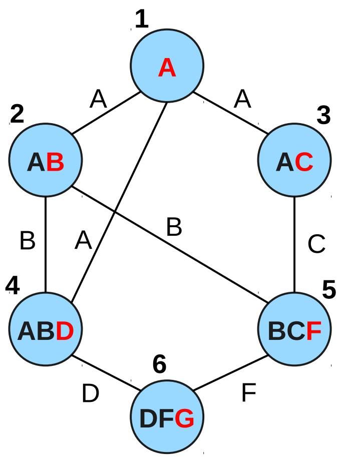
D	F	G	P(G|D,F)
1	2	3	1
2	1	3	1
…	…	…	0

Belief network

A
1
2
3

A	B
1	2
1	3
2	1
2	3
3	1
3	2

A	C
1	2
3	2



A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

B	C	F
1	2	3
3	2	1

D	F	G
1	2	3
2	1	3

Flat constraint network

IBP – inference power for zero beliefs

- **Theorem:**

  Trace of zero beliefs of Iterative Belief Propagation =

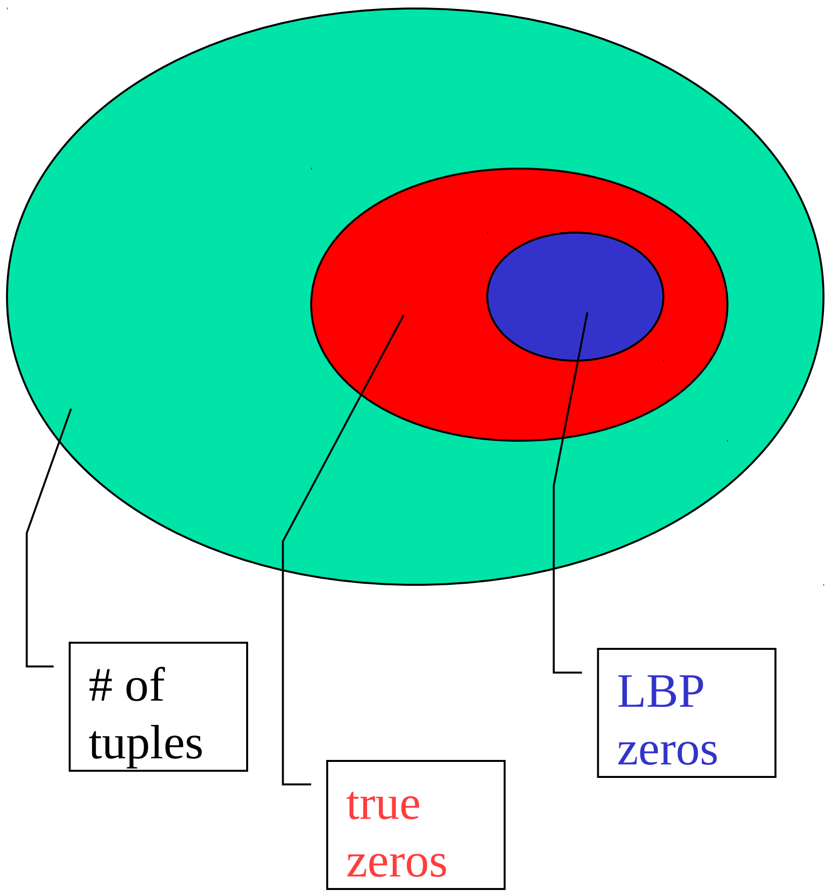
  Trace of invalid tuples of arc-consistency on flat network

- **Soundness:**
  - The inference of zero beliefs by IBP converges in a finite number of iterations
  - all the inferred zero beliefs are correct

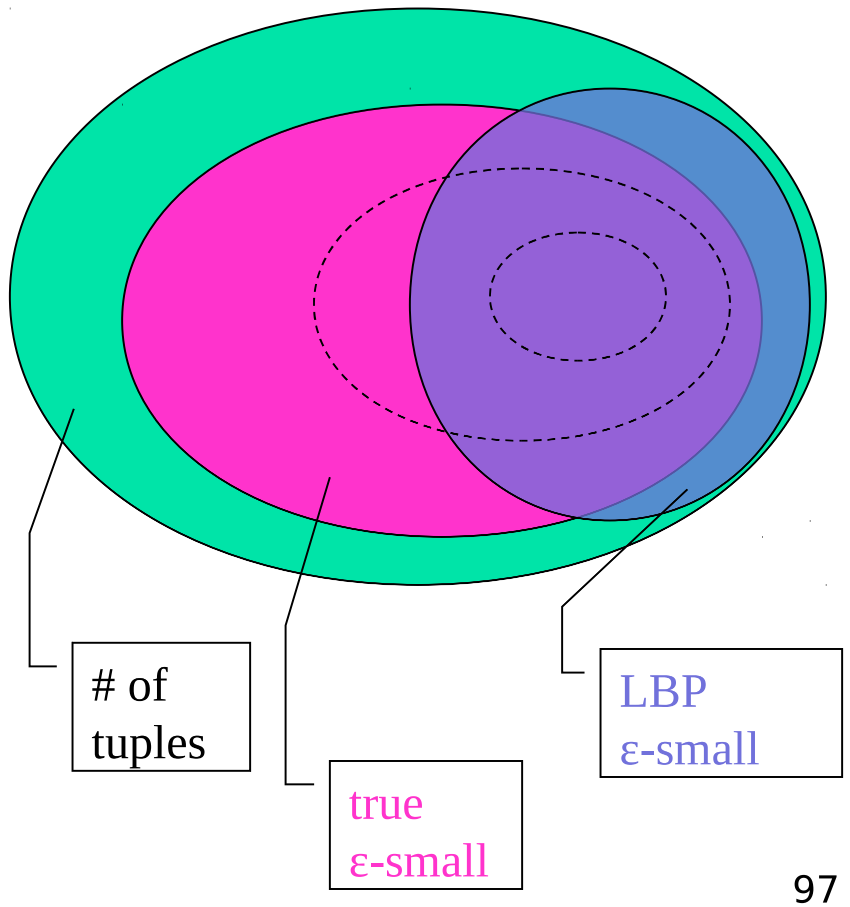
- **Incompleteness:**
  - IBP may not infer all the true zero beliefs

Zero and ε-Small Beliefs

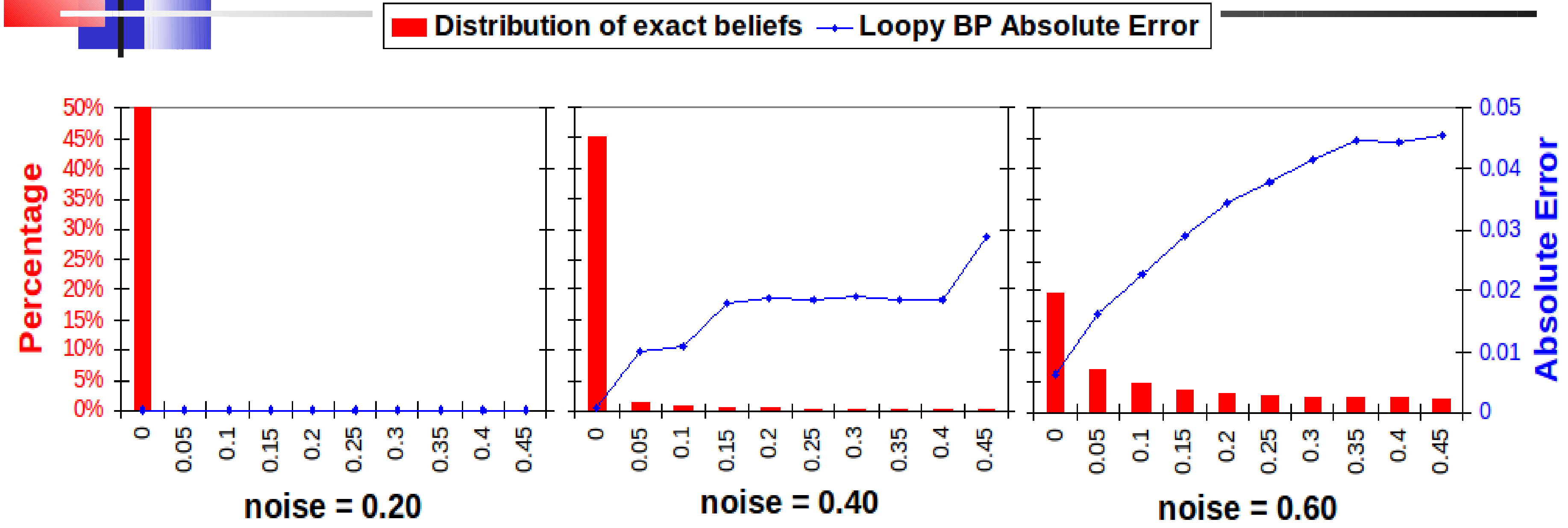
Zero beliefs



ε-small beliefs

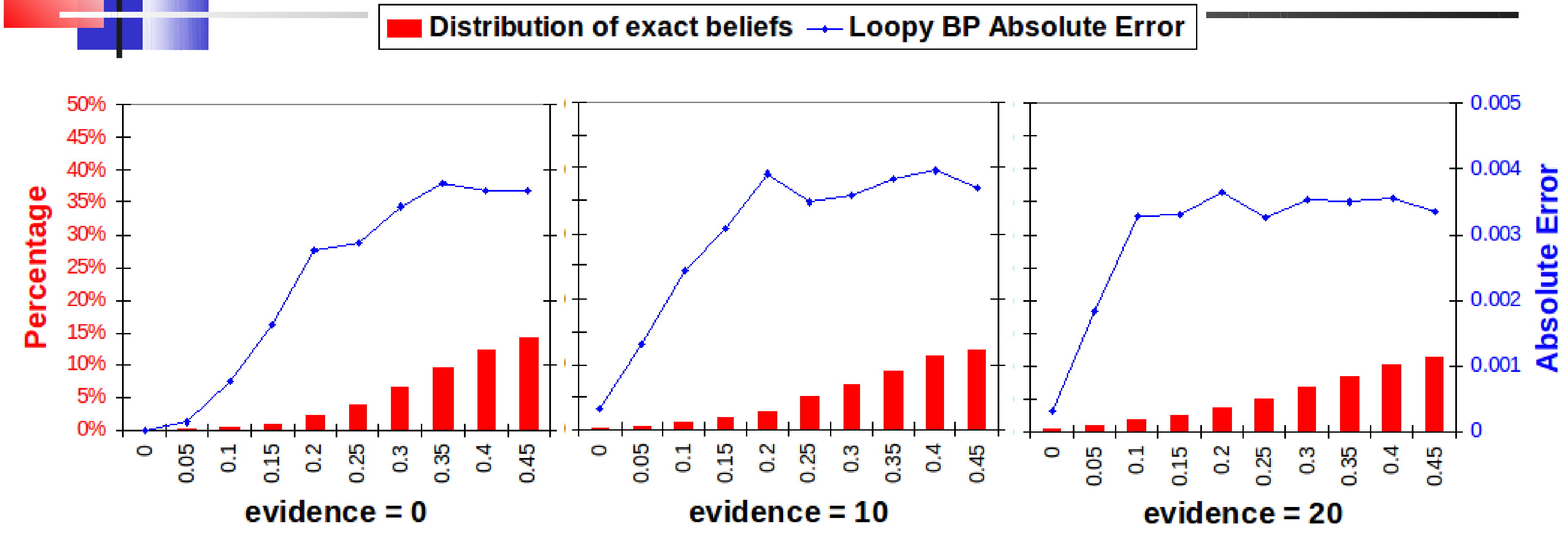


Coding Networks



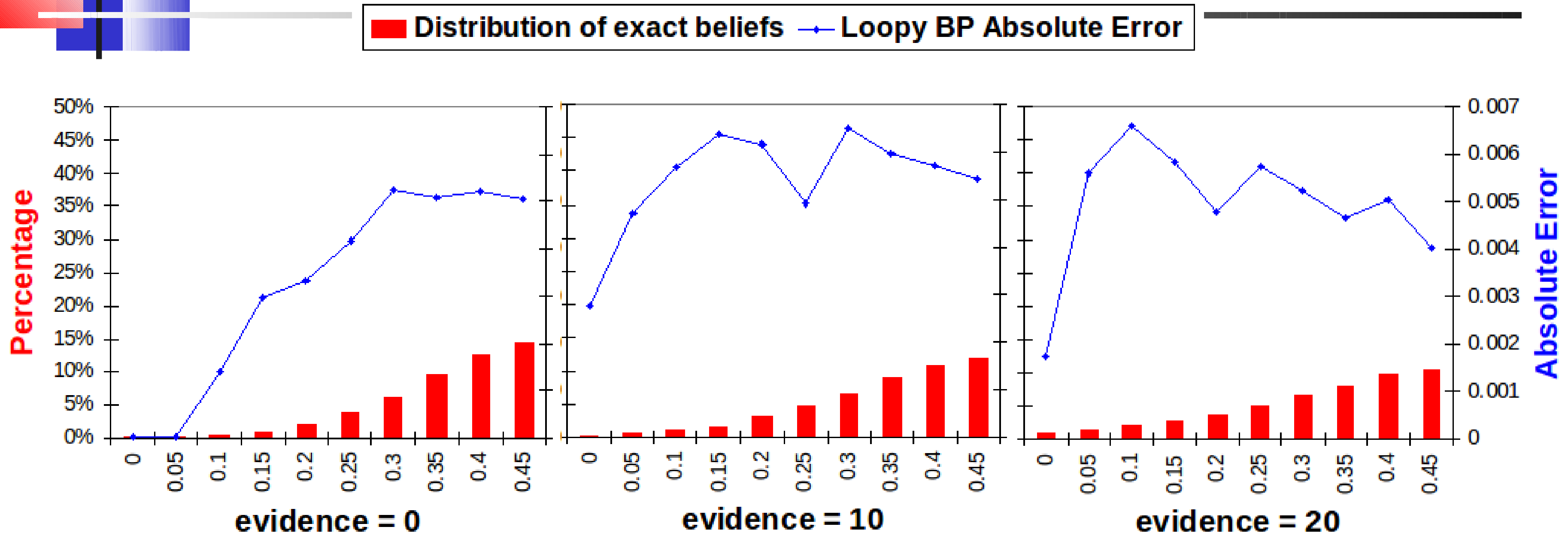
N=200, 1000 instances, treewidth=15

10x10 Grids



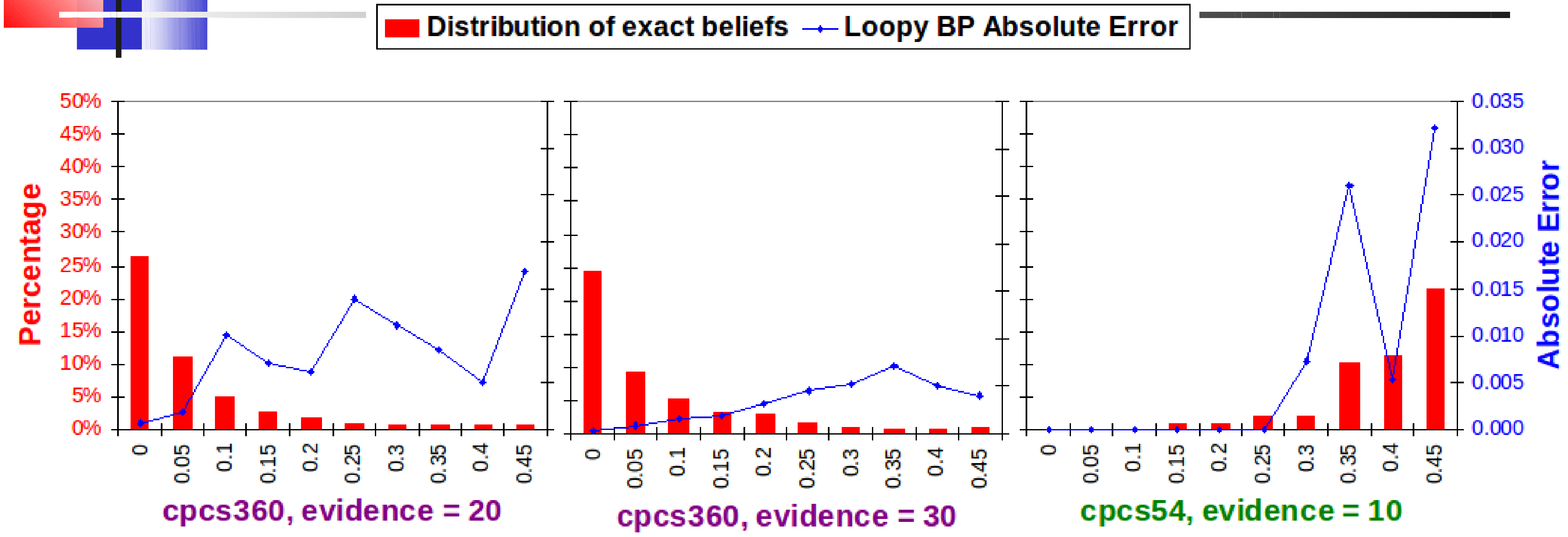
N=100, 100 instances, w*=15

Random Networks



N=80, 100 instances, w*=15

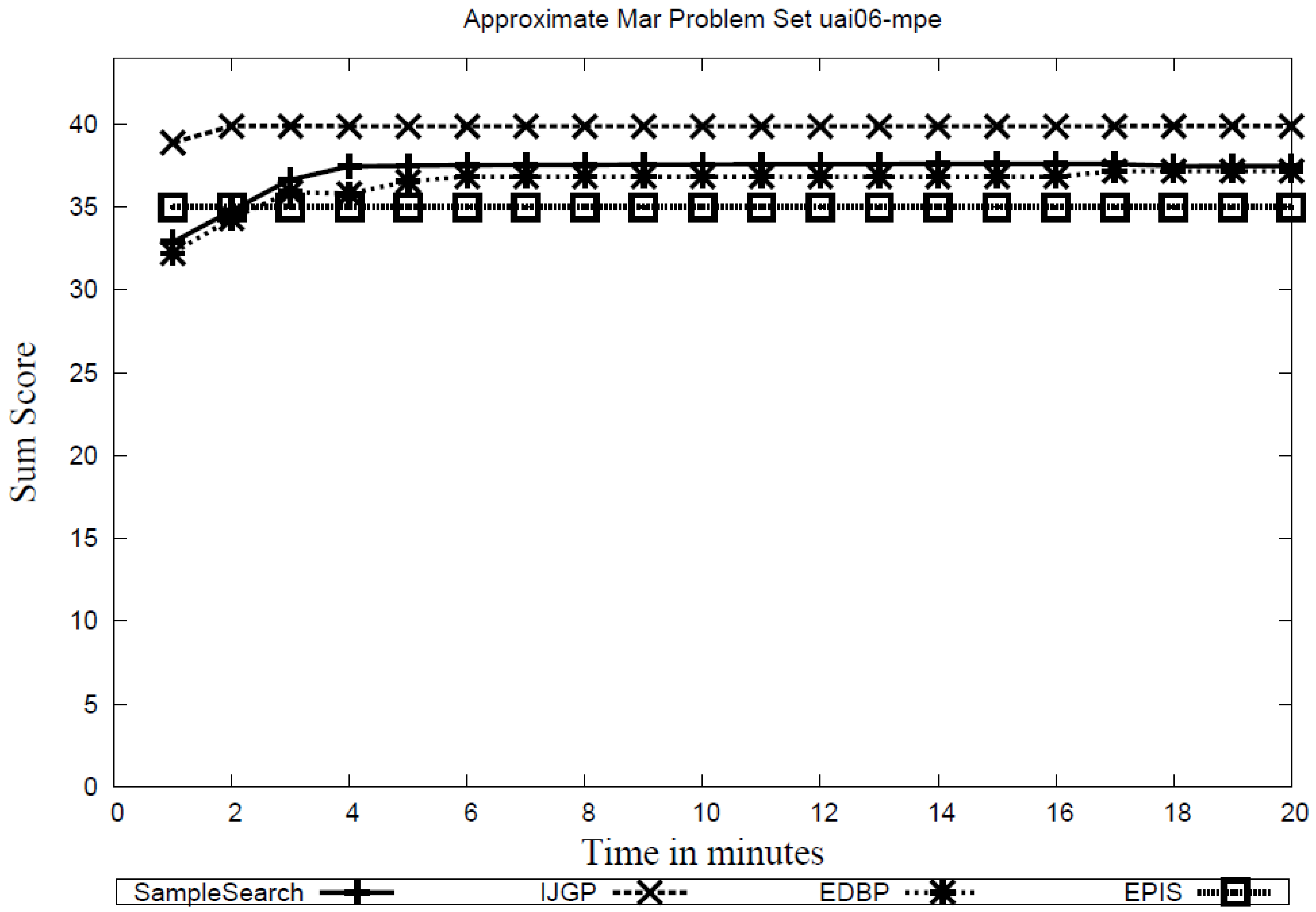
CPCS 54, CPCS360



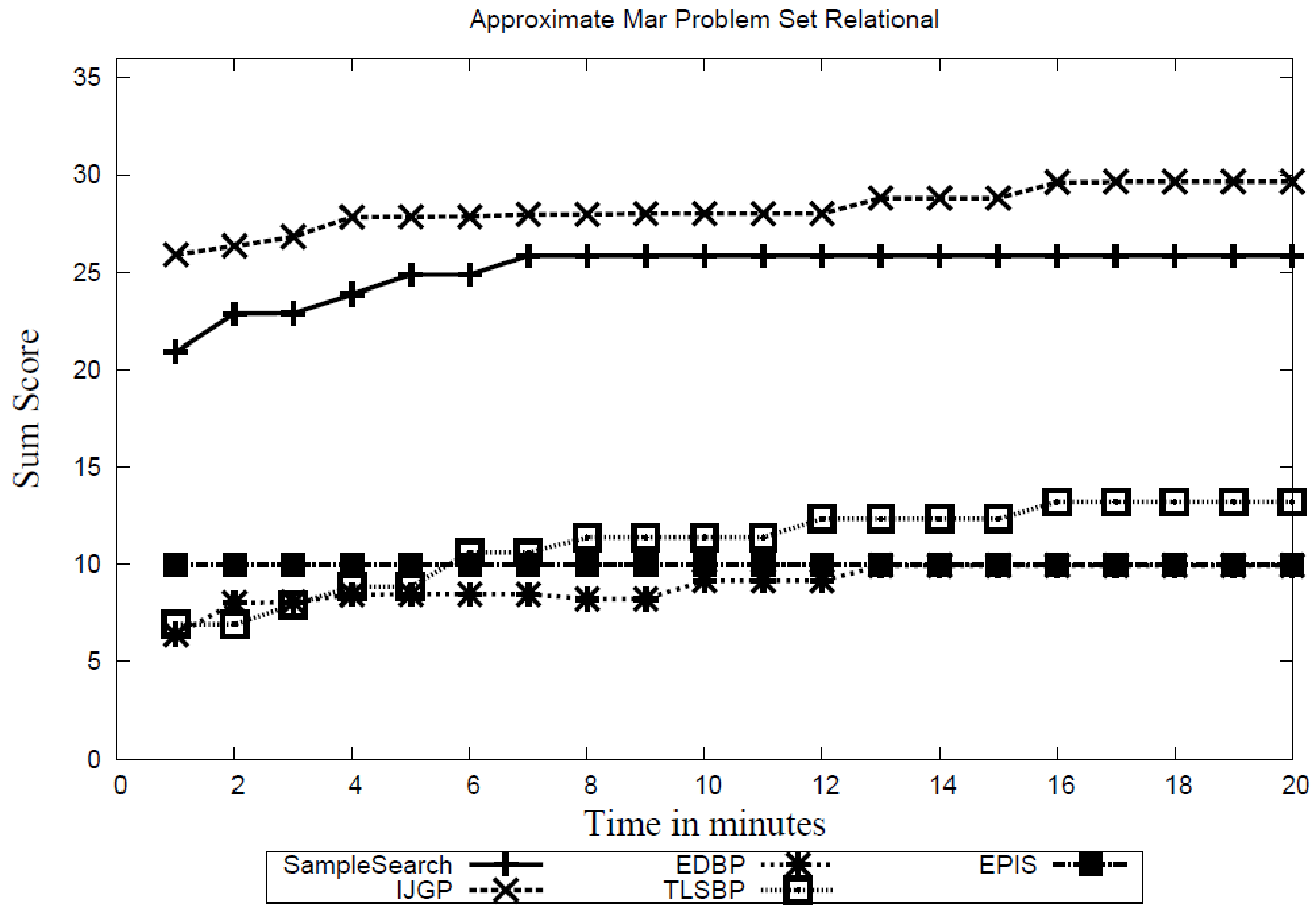
CPCS360: 5 instances, w*=20

CPCS54: 100 instances, w*=15

IJGP on UAI06 problems

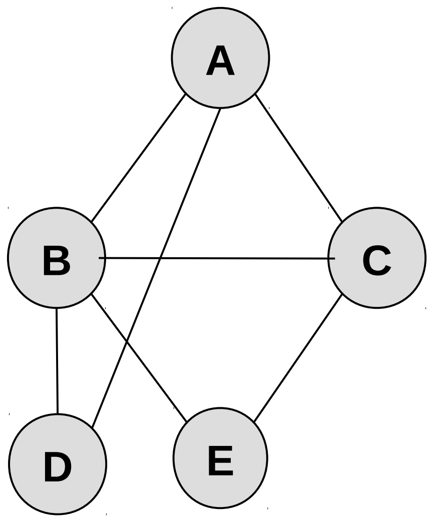


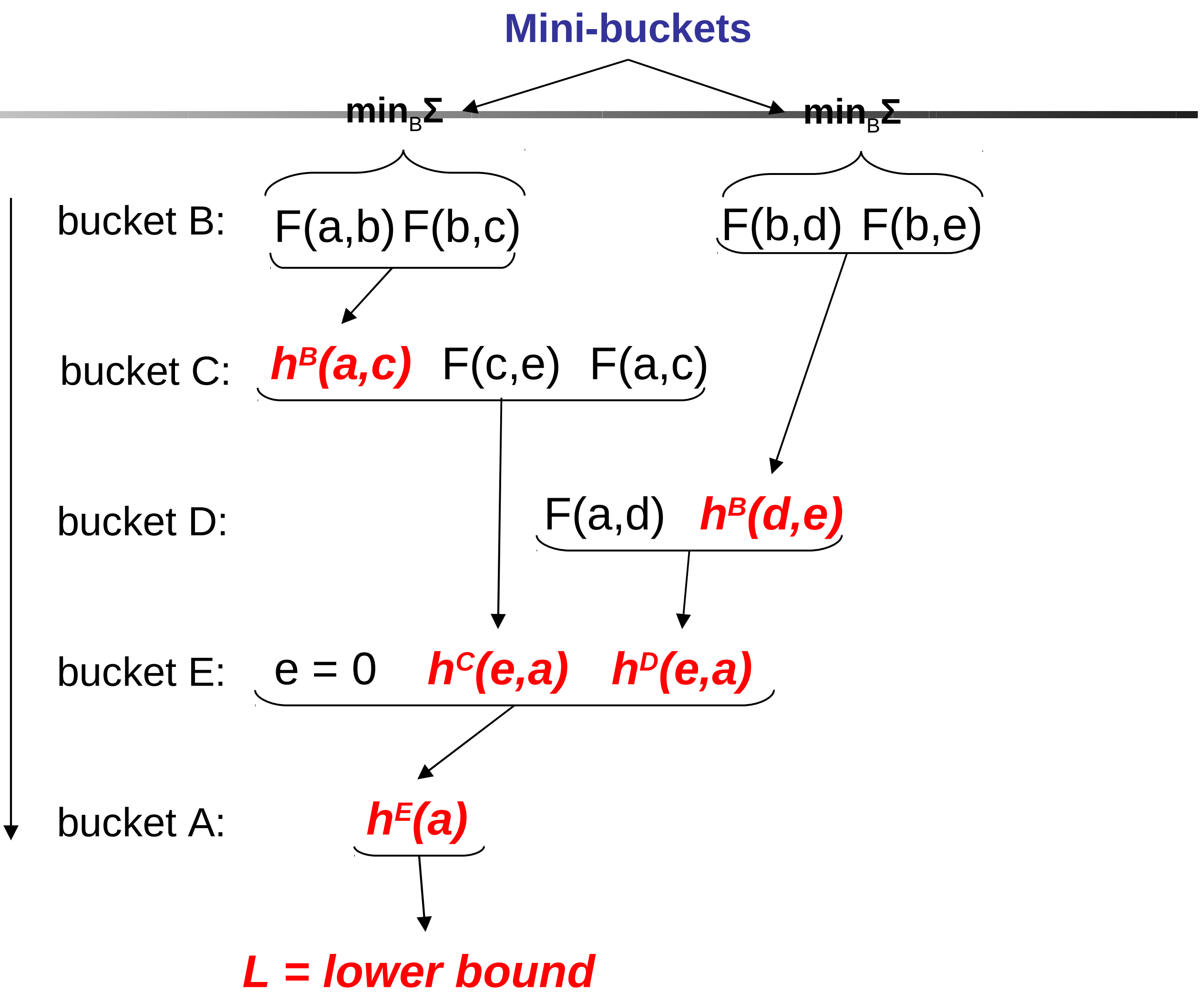
IJGP on Set Relational



Using Mini-bucket approximation in search

Mini-Bucket can be used to guide more than one solution




Mini-buckets
$\min_B \Sigma$
$\min_B \Sigma$
bucket B: F(a,b) F(b,c)   F(b,d) F(b,e)
bucket C: $h^B(a,c)$ F(c,e) F(a,c)
bucket D: F(a,d) $h^B(d,e)$
bucket E: e = 0 $h^C(e,a)$ $h^D(e,a)$
bucket A: $h^E(a)$
L = lower bound

Basic Heuristic Search Schemes

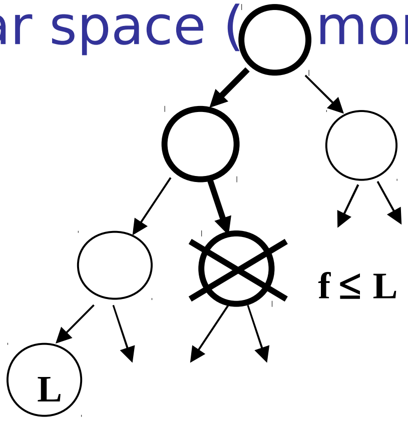
Heuristic function **$f(x^p)$** computes a lower bound on the best

extension of **x^p** and can be used to guide a heuristic search algorithm. We focus on:

1. Branch-and-Bound

Use heuristic function **$f(x^p)$** to prune the depth-first search tree

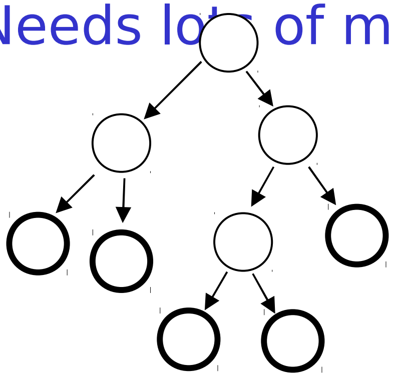
Linear space (more)



2. Best-First Search

Always expand the node with the highest heuristic value **$f(x^p)$**

Needs lo of memory



Heuristic search

- Mini-buckets record upper-bound heuristics
- The evaluation function over $\overline{\mathrm{x}}_{\mathrm{p}} = (x_1, \ldots x_p)$

$$f(\bar{x}_p) = g(\bar{x}_p) h(\bar{x}_p)$$

$$g(\bar{x}_p) = \prod_{i=1}^{p-1} P(x_i \mid pa_i)$$

$$h(\bar{x}_p) = \prod_{h_j \in bucket_p} h_j$$

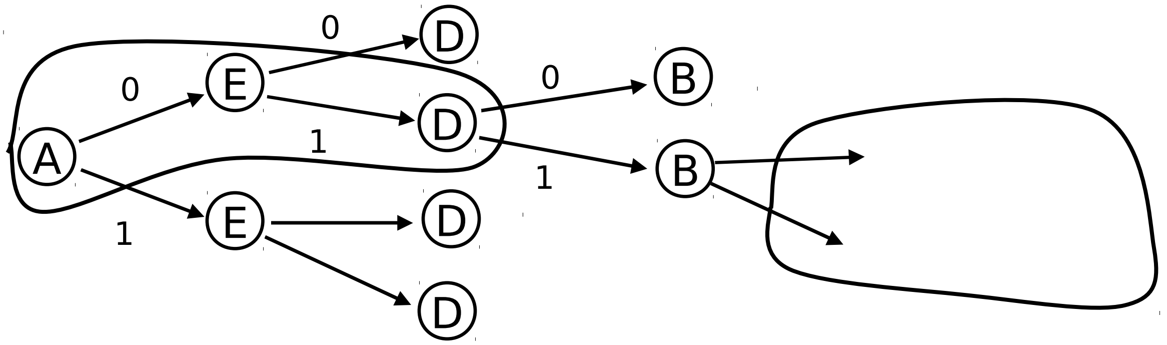
- **Best-first:** expand a node with maximal evaluation function
- **Branch and Bound:** prune if f <= upper bound
- **Properties:**
    - an exact algorithm
    - Better heuristics lead to more pruning

Heuristic Function

Given a cost function

$$P(a,b,c,d,e) = P(a) \bullet P(b|a) \bullet P(c|a) \bullet P(e|b,c) \bullet P(d|b,a)$$

Define an evaluation function over a partial assignment as the probability of it's best extension

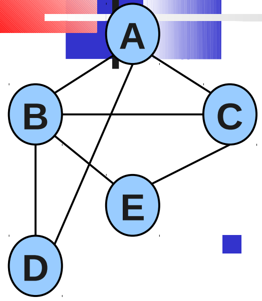


$$f^*(a,e,d) = \max_{b,c} P(a,b,c,d,e) =$$

$$= \underbrace{P(a)} \bullet \max_{b,c} \underbrace{P(b|a) \bullet P(c|a) \bullet P(e|b,c) \bullet P(d|a,b)}$$

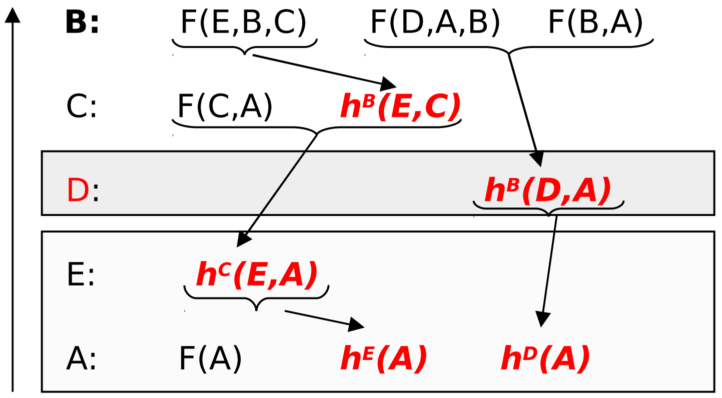
$$= g(a,e,d) \bullet H^*(a,e,d)$$

MBE Heuristics

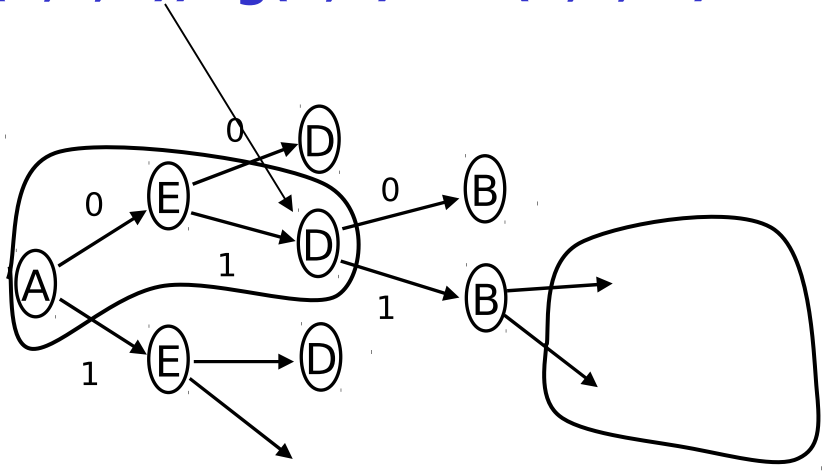


Cost Network

- Given a partial assignment $\mathbf{x^p}$, estimate the cost of the best extension to a full solution
- The evaluation function $\mathbf{f(x^p)}$ can be computed using function recorded by the Mini-Bucket scheme



f(a,e,D))=g(a,e) + H(a,e,D)

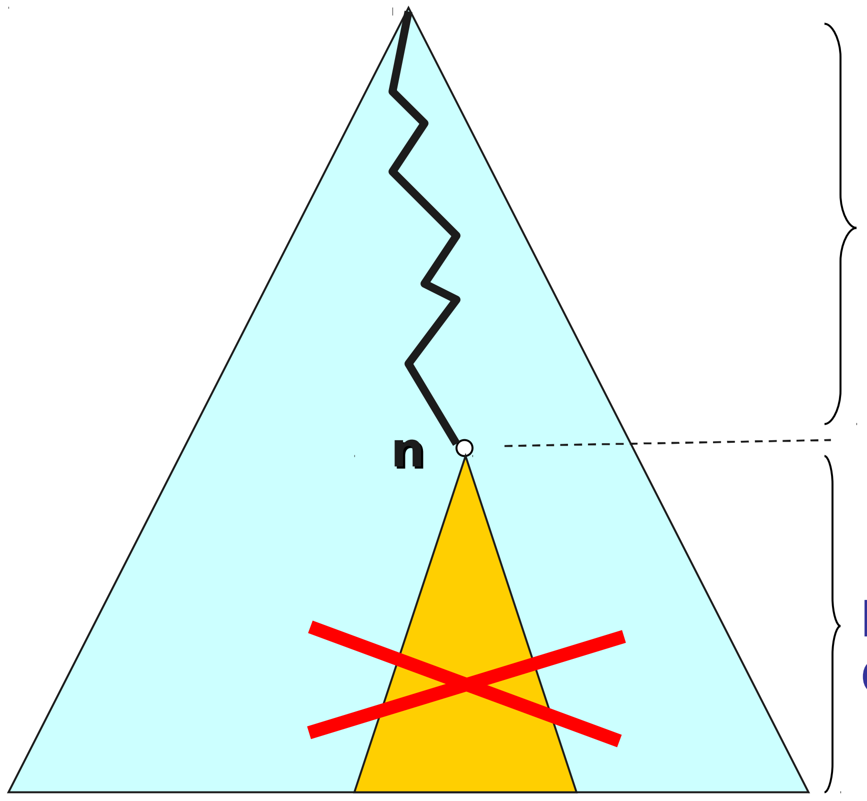


$$\mathbf{f(a,e,D)} = \underbrace{F(a)}_{g} + \underbrace{h^B(D,a) + h^C(e,a)}_{h \text{ - is admissible}}$$

Properties

- Heuristic is consistent/monotone
- Heuristic is admissible
- Heuristic is computed in linear time
- IMPORTANT:
  - Mini-buckets generate heuristics of varying strength using control parameter – bound i
  - Higher bound -> more preprocessing -> stronger heuristics -> less search
  - Allows controlled trade-off between preprocessing and search

Classic Branch-and-Bound



OR Search Tree

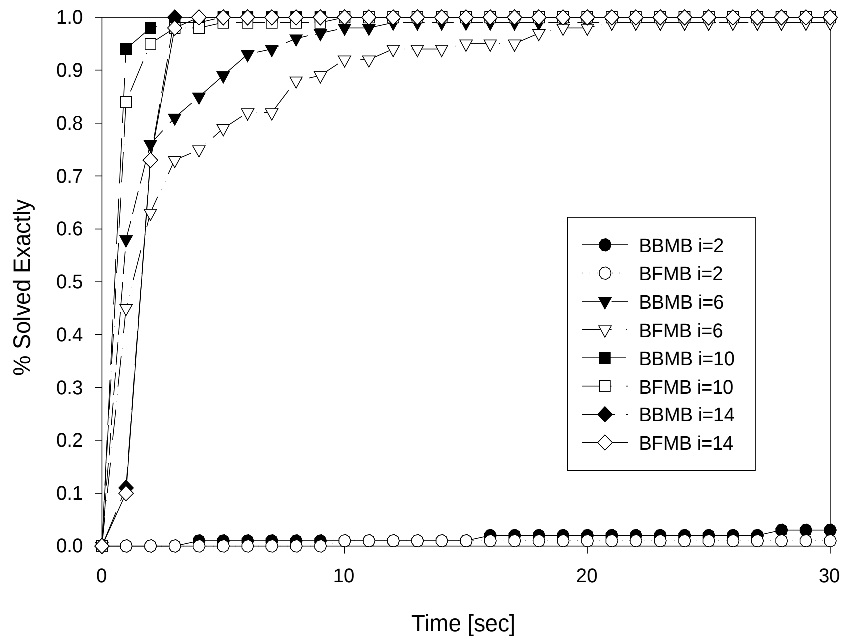
Upper Bound **UB**

Lower Bound **LB**

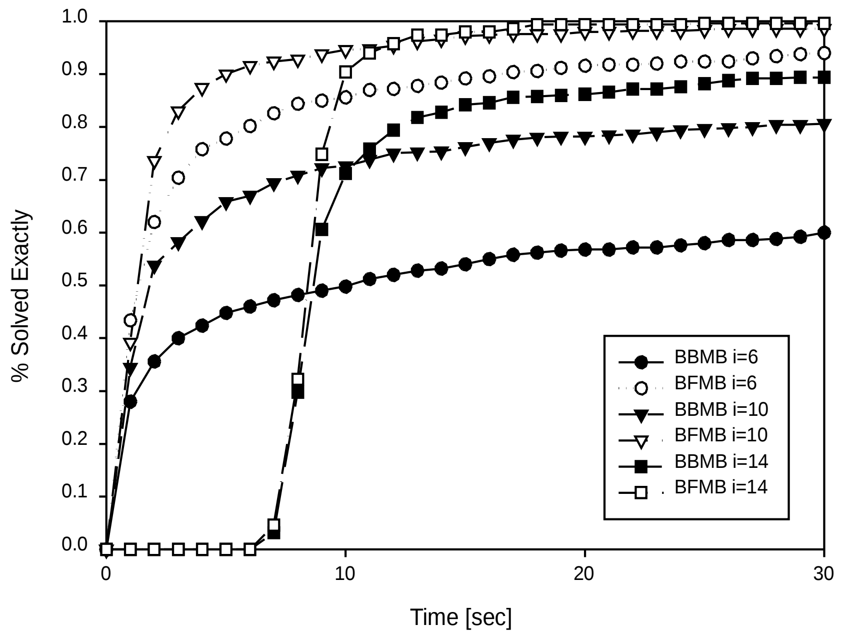
LB(n) = g(n) + h(n)

Prune if LB(n) ≥ UB

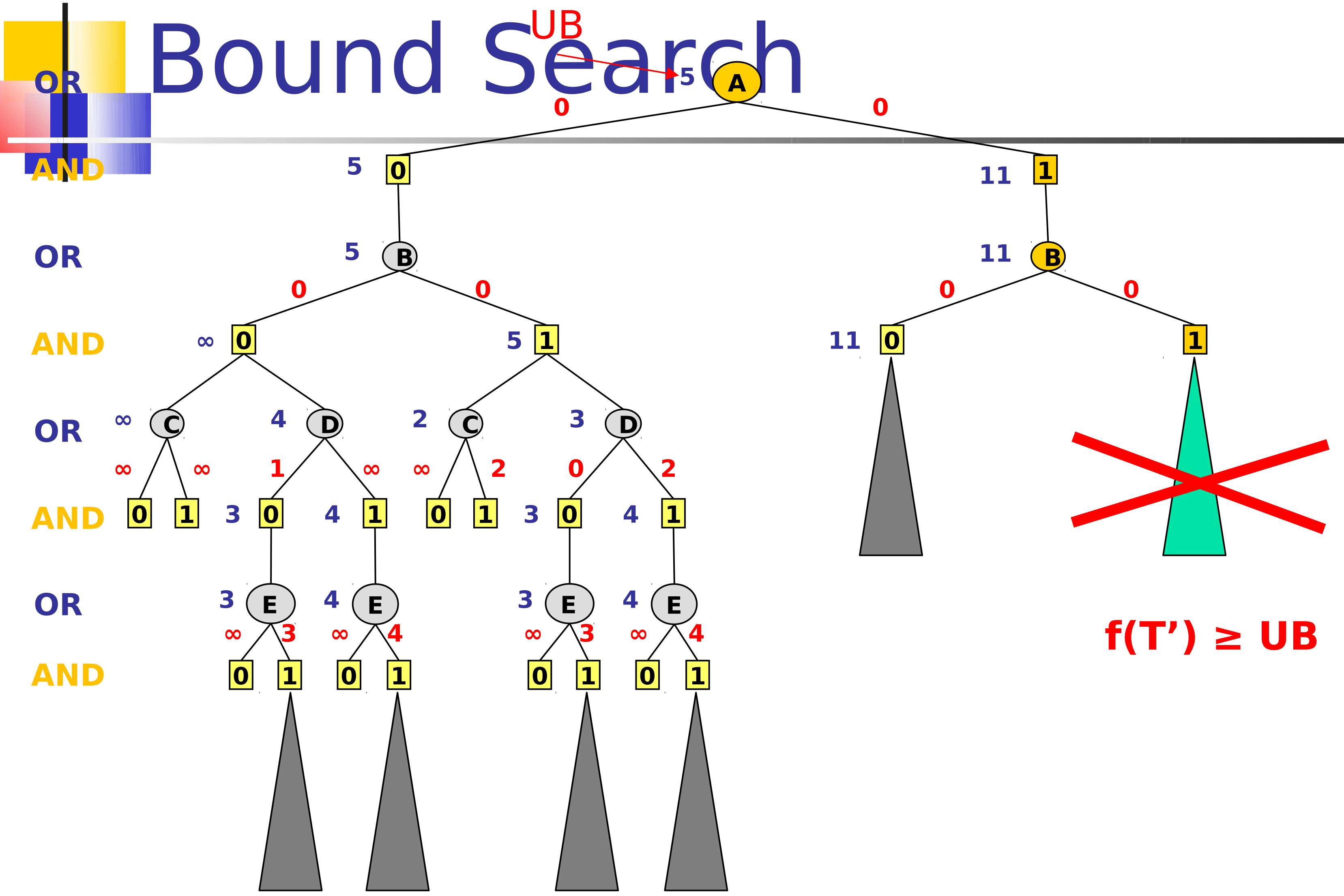
Empirical Evaluation of mini-bucket heuristics



Random Coding, K=100, noise 0.32

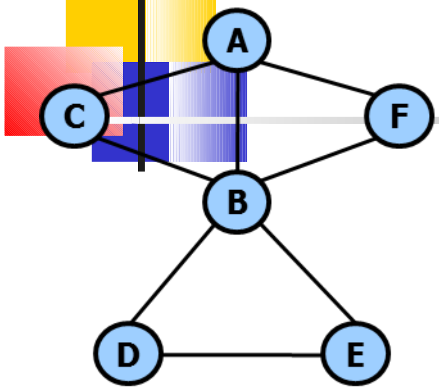


AND/OR Branch-and-Bound Search



f(T') ≥ UB

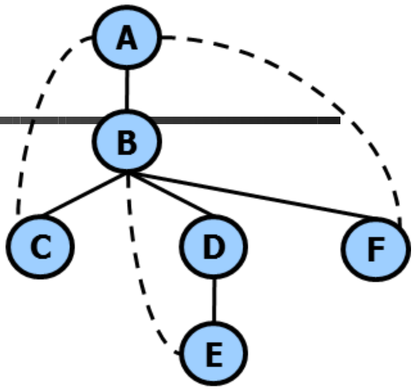
Heuristic Evaluation Function

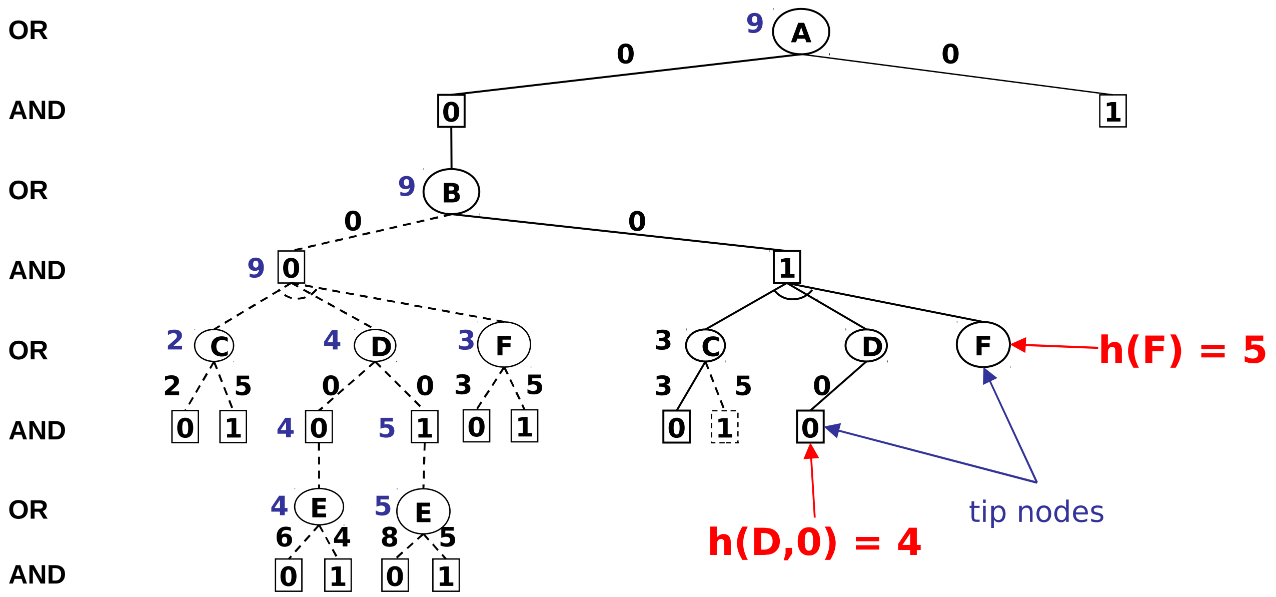


A	B	C	$f_1(ABC)$
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

A	B	F	$f_2(ABF)$
0	0	0	3
0	0	1	5
0	1	0	1
0	1	1	4
1	0	0	6
1	0	1	5
1	1	0	6
1	1	1	5

B	D	E	$f_3(BDE)$
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4





$f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 \leq f^*(T$

Software & Competitions

- **How to use the software**
  - http://graphmod.ics.uci.edu/group/Software
  - http://mulcyber.toulouse.inra.fr/projects/toulbar2
- **Reports on competitions**
  - UAI-2006, 2008, 2010 Competitions
    - PE, MAR, MPE tasks
  - CP-2006 Competition
    - WCSP task

Toulbar2 and aolib

- toulbar2

  http://mulcyber.toulouse.inra.fr/gf/project/toulbar2

  (Open source WCSP, MPE solver in C++)

- aolib

  http://graphmod.ics.uci.edu/group/Software

  (WCSP, MPE, ILP solver in C++, inference and counting)

- Large set of benchmarks

  http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP

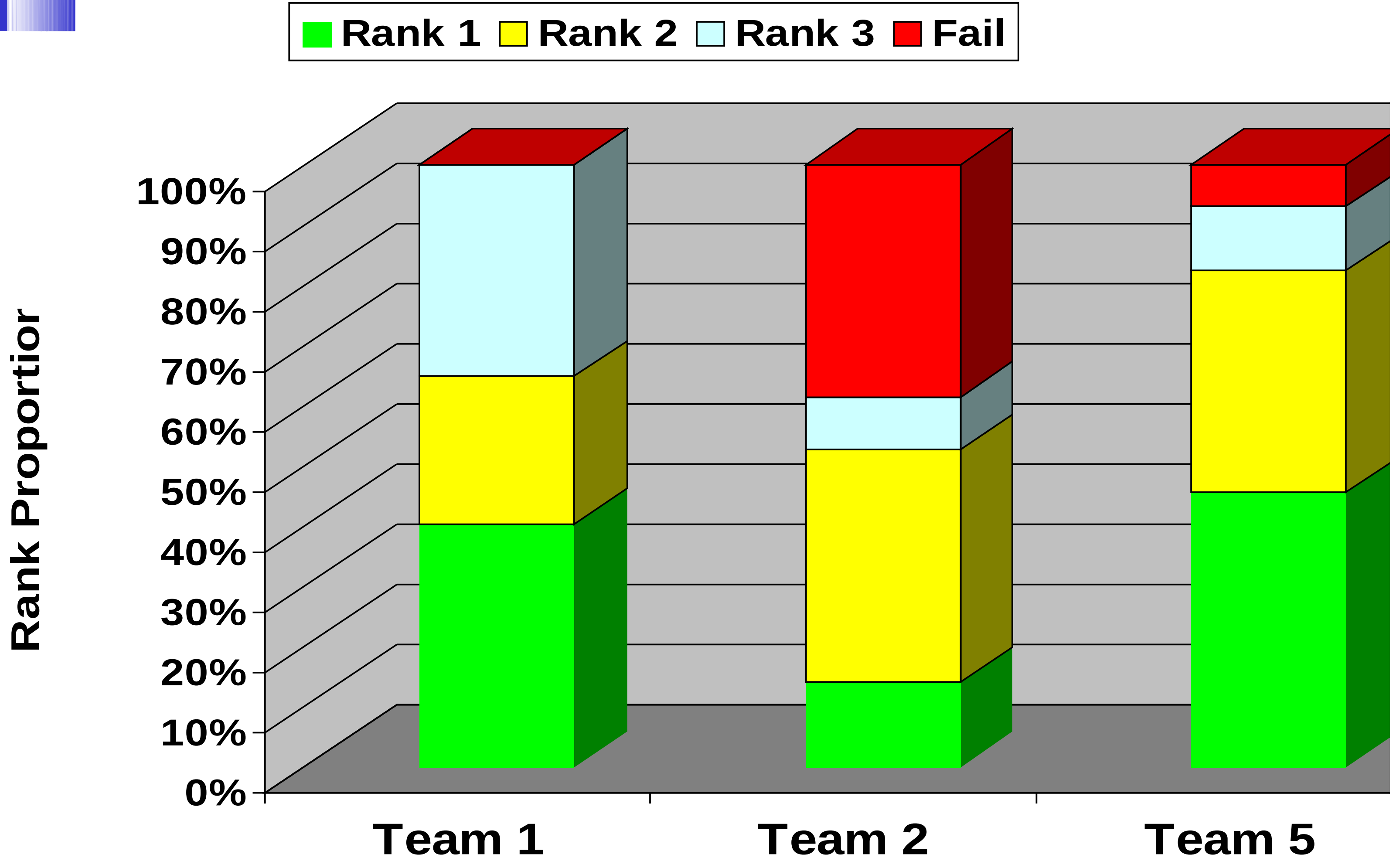
  http://graphmod.ics.uci.edu/group/Repository

UAI-2006 Competition

- **Team 1 (UCLA)**
  - David Allen, Mark Chavira, Arthur Choi, Adnan Darwiche
- **Team 2 (IET)**
  - Masami Takikawa, Hans Dettmar, Francis Fung, Rick Kissh
- **Team 5 (UCI)**
  - Radu Marinescu, Robert Mateescu, Rina Dechter
  - Used **AOBB-C+SMB(i)** solver for MPE

UAI-2006 Results

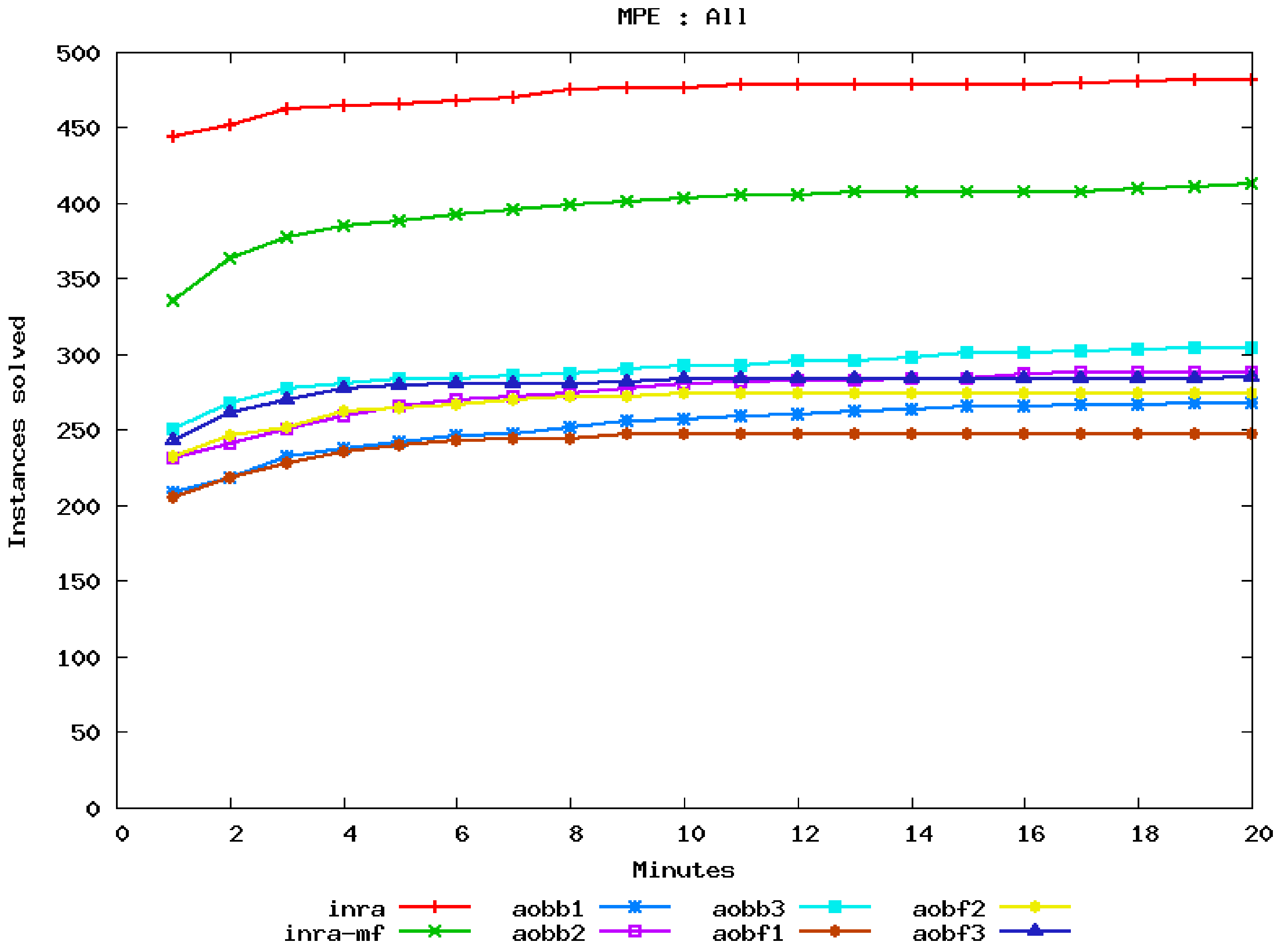
Rank Proportions (how often was each team a particular rank, rank 1 is best



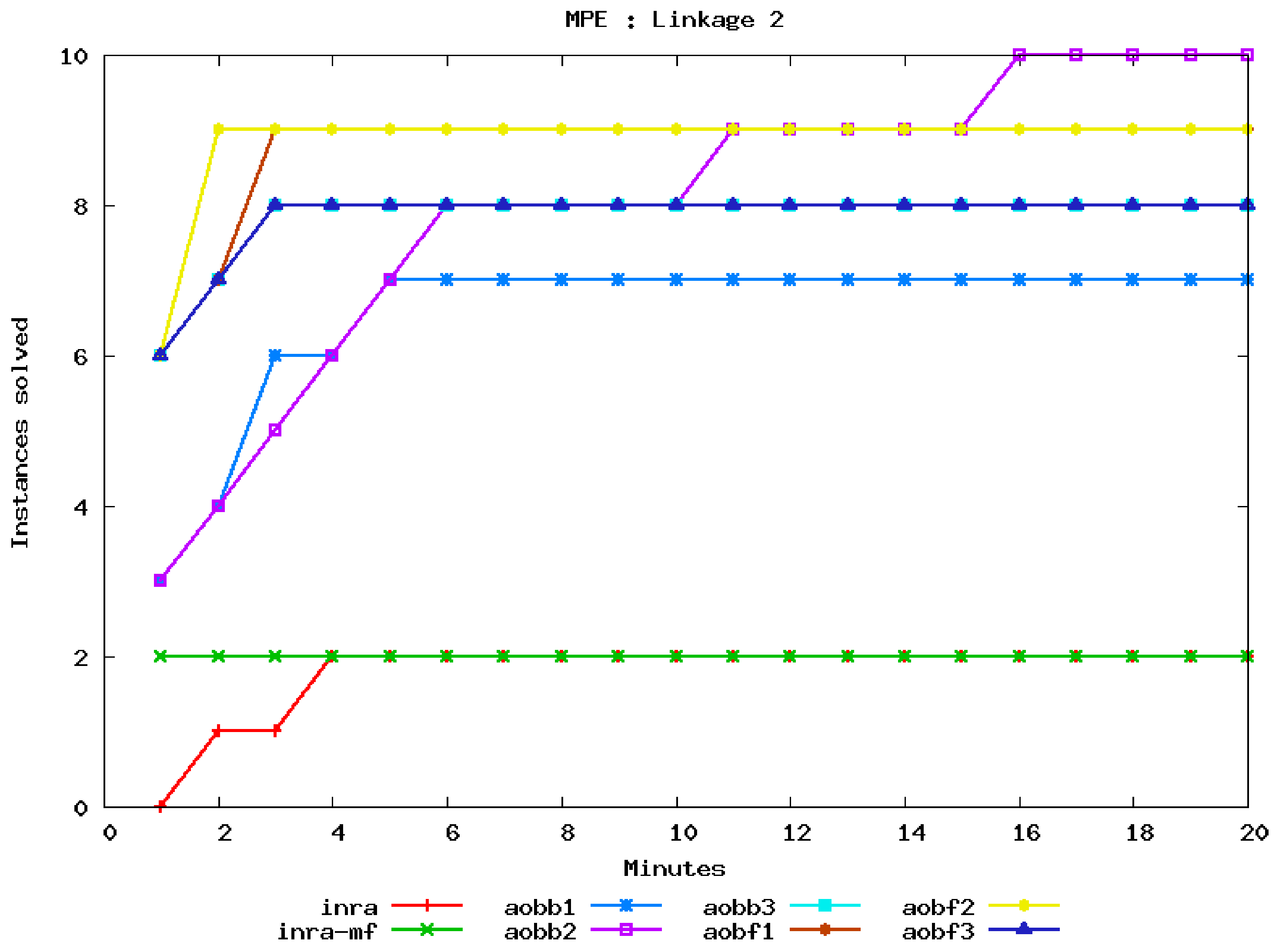
UAI-2008 Competition

- **AOBB-C+SMB(i) – (i = 18, 20, 22)**
  - AND/OR Branch-and-Bound with pre-compiled mini-bucket heuristics (i-bound), full caching, static pseudo-trees, constraint propagation
- **AOBF-C+SMB(i) – (i = 18, 20, 22)**
  - AND/OR Best-First search with pre-compiled mini-bucket heuristics (i-bound), full caching, static pseudo-trees, no constraint propagation
- **Toulbar2**
  - OR Branch-and-Bound, dynamic variable/value orderings, EDAC consistency for binary and ternary cost functions, variable elimination of small degree (2) during search
- **Toulbar2/BTD**
  - DFBB exploiting a tree decomposition (AND/OR), same search inside clusters as toulbar2, full caching (no cluster merging), combines RDS and EDAC, and caching lower bounds

UAI-2008 Results



UAI-2008 Results (contd.)



UAI-2010 Competition

- Tasks
  - PR: probability of evidence
  - MAR: posterior marginals
  - MPE: most probable explanation
- 3 tracks: 20 sec, 20 min, 1 hour
  - PR, MAR - 204 instances; MPE - 442 instances
    - CSP, grids, image alignment, medical diagnosis, object detection, pedigree, protein folding, protein-protein interaction, relational model, segmentation
- Exact and approximate solvers

UAI-2010 Results

(Mateescu et al, JAIR2010), (Dechter et al, UAI2002)

- MAR task
  - **1st place** (20 min, 1 hour) – (impl. by Vibhav Gogate)
  - Anytime **IJGP(i**) with randomized orderings and SAT based domain pruning
- PR task

(Gogate, Domingos and Dechter UAI2010)

  - **1st place** (20 min, 1 hour) – (impl. by Vibhav Gogate)
  - Formula **SampleSearch** with IJGP(3) based importance distribution, w-cutset sampling, minisat based search, rejection control
- MPE task
  - **3rd place** (all tracks) – (impl. by Lars Otten)
  - **AND/OR BnB** with mini-buckets, randomized min-fill based pseudo tree, LDS based search for initial upper bound

(Marinescu and Dechter, AIJ2009), (Otten and Dechter, ISAIM2010)